UNIVERSITÀ DEGLI STUDI DI PAVIA
ISTITUTO DI MATEMATICA

M. ITALIANI G. SERAZZI

LA PROGRAMMAZIONE DEL CALCOLATORE C.N.D.

UNIVERSITÀ DEGLI STUDI DI PAVIA
ISTITUTO DI MATEMATICA

M. ITALIANI G. SERAZZI

LA PROGRAMMAZIONE DEL CALCOLATORE C.N.D.

PREFAZIONE

Nel presente volume è stata raccolta la parte delle lezioni del corso di 'Teoria e Applicazione delle Macchine Calcolatrici' relativa alla struttura ed alla programmazione del Calcolatore Didattico CND.

Le parti del corso riguardanti i concetti introduttivi alla elaborazione automatica dei dati, i linguaggi simbolici di programmazione e le nozioni generali sul 'software' verranno raccolte in altri volumi.

PAVIA, marzo 1971

INDICE

INTRODUZIONE

1. ORGANIZZAZIONE DEL CALCOLATORE

1.1 CONFIGURAZIONE 2
1.2 LA MEMORIA 3
1.3 RAPPRESENTAZIONE INTERNA DELLE INFORMAZIONI 4
1.4 L'UNITA' DI GOVERNO 9
1.5 IL QUADRO DI COMANDO 12
1.6 LA UNITA' DI INPUT-OUTPUT 19
1.7 SEQUENZE OPERATIVE PER L'UTILIZZAZIONE DEL CALCOLATORE . 23
1.8 LA ZONA RISERVATA DELLA MEMORIA 27
1.9 IL REGISTRO STATO DEL PROGRAMMA 30

2. IL LINGUAGGIO MACCHINA

2.1 GENERALITA' 34
2.2 FORMATI DELLE ISTRUZIONI 37
2.3 INTERPRETAZIONE ED ESECUZIONE DELLE ISTRUZIONI 42
2.4 REDAZIONE DI UN PROGRAMMA 47

3. DESCRIZIONE DELLE ISTRUZIONI

3.1 ISTRUZIONI ARITMETICHE E DI CONFRONTO 52
3.2 ISTRUZIONI DI INPUT-OUTPUT 82
3.3 ISTRUZIONI DI CONTROLLO 93

4. TECNICHE DI PROGRAMMAZIONE

4.1 ANALISI DI UN PROBLEMA E DIAGRAMMA DI FLUSSO 104
4.2 STRUTTURA DEI PROGRAMMI IN LINGUAGGIO MACCHINA 108
4.3 MODIFICA DELLE ISTRUZIONI 113
4.4 ESEMPI DI PROGRAMMI IN LINGUAGGIO MACCHINA 116

5. SOTTOPROGRAMMI

5.1 GENERALITA' 154
5.2 IL RIENTRO DA UN SOTTOPROGRAMMA 158
5.3 I PARAMETRI DI UN SOTTOPROGRAMMA 160
5.4 UN ESEMPIO DI SOTTOPROGRAMMA 162

6. ESTENSIONE DEL CALCOLATORE C.N.D.

6.1 I REGISTRI INDICE 168
6.2 L'INDIRIZZAMENTO INDIRETTO DELLA MEMORIA 170
6.3 LE ISTRUZIONI PER I REGISTRI INDICE 173
6.4 USO DEI REGISTRI INDICE NEI SOTTOPROGRAMMI 189
6.5 RICOLLOCABILITA' DEI PROGRAMMI 203

7. IL LINGUAGGIO ASSEMBLATIVO C.N.D.

7.1 INTRODUZIONE 208
7.2 ISTRUZIONI DEL LINGUAGGIO ASSEMBLATIVO C.N.D. 211
7.3 PROGRAMMI IN LINGUAGGIO ASSEMBLATIVO 219

APPENDICE 229

INTRODUZIONE

Il presente volume descrive un ipotetico calcolatore numerico al quale è stato assegnato il nome convenzionale CND (calcolatore numerico didattico).

Il calcolatore CND, pur avendo una organizzazione piuttosto semplice, ha tuttavia le caratteristiche fondamentali dei moderni elaboratori elettronici e pertanto la sua conoscenza può notevolmente facilitare l'apprendimento di qualsiasi sistema per l'elaborazione dei dati.

Inoltre il repertorio delle istruzioni del calcolatore CND è molto limitato e quindi il tempo richiesto per acquisire le nozioni necessarie per la redazione di un programma risulta ragionevolmente breve.

Come si vedrà la descrizione delle istruzioni è fatta al livello del linguaggio macchina e non mediante l'uso di un linguaggio simbolico, che tuttavia viene introdotto nel cap. 7.
Si è infatti ritenuto che la conoscenza delle istruzioni macchina sia indispensabile per comprendere i fondamenti della programmazione e costituisce una necessaria premessa per una trattazione degli argomenti che verranno trattati in seguito.

Nella redazione del volume si è supposto che il lettore fosse familiare con alcuni concetti elementari dell'elaborazione dei dati quali i principi della numerazione binaria ed esadecimale e le nozioni di memoria e di programma.

CAPITOLO 1

ORGANIZZAZIONE DEL CALCOLATORE

1.1 CONFIGURAZIONE

1.2 LA MEMORIA

1.3 RAPPRESENTAZIONE INTERNA DELLE INFORMAZIONI

1.4 L'UNITA' DI GOVERNO

1.5 IL QUADRO DI COMANDO

1.6 LA UNITA' DI INPUT-OUTPUT

1.7 SEQUENZE OPERATIVE PER L'UTILIZZAZIONE DEL CALCOLATORE

1.8 LA ZONA RISERVATA DELLA MEMORIA

1.9 IL REGISTRO STATO DEL PROGRAMMA

1.1 CONFIGURAZIONE

Il calcolatore CND, nella sua configurazione base, è composto dalle seguenti unità :

a) Unità di memoria a nuclei magnetici in grado di memorizzare le informazioni da elaborare e il programma di elaborazione

b) Unità di governo che interpreta le istruzioni componenti il programma registrato nella unità di memoria e provvede alla loro esecuzione.

c) Quadro di comando (console) contenente una tastiera esadecimale per l'introduzione dei programmi e una serie di tasti e lampade che permettono il controllo dell'intero sistema e l'esecuzione delle operazioni per l'avviamento o l'arresto delle elaborazioni.

d) Unità di input consistente in una tastiera che contiene i 64 caratteri componenti l'alfabeto del calcolatore.

e) Unità di output consistente in una stampante seriale in grado di stampare i 64 caratteri dell'alfabeto.

L'unità di input e l'unità di output sono in realtà integrate in una telescrivente e la stampa può essere provocata sia manualmente dall'operatore che opera sulla tastiera, sia automaticamente dal calcolatore che trasmette all'unità di stampa dati registrati nella memoria a nuclei.

1.2 LA MEMORIA

Il calcolatore CND è dotato di una memoria costituita da 4096 gruppi, di 9 nuclei magnetici ciascuno.

Ogni nucleo è in grado di immagazzinare una informazione binaria (bit) che può assumere i valori convenzionali 0 ed 1 ed ogni gruppo di nove nuclei costituisce una "cella" di memoria che viene usualmente chiamata "byte".

Uno dei nove bits che possono essere immagazzinati in un byte viene utilizzato per il controllo di disparità (il numero di "bit 1" presenti in un byte deve essere sempre dispari) mentre gli altri otto danno luogo a $2^8 = 256$ configurazioni diverse che possono essere interpretate, come si vedrà, in vario modo.

A ciascuno dei 4096 bytes costituenti la memoria viene associato un indirizzo consistente in un numero esadecimale compreso tra 0 e FFF (cioè tra 0 e il numero decimale 4095).

Non è invece possibile associare un indirizzo a parti di byte e in particolare ad un singolo bit.

In generale una informazione registrata nella memoria occupa più bytes; tali bytes sono consecutivi e disposti per indirizzi crescenti nel senso che la parte più significativa dell'informazione è quella con indirizzo minore mentre la meno significativa è quella con indirizzo maggiore.

Come indirizzo di una informazione che occupa più di un byte si assume quello del byte di indirizzo inferiore (spesso si dice il byte di sinistra, facendo la convenzione che le informazioni siano scritte da sinistra verso destra). Così, ad esempio, l'indirizzo della informazione che occupa i bytes compresi tra quello di indirizzo 2A3 (675 decimale) e quello di indirizzo 2A7 (679 decimale) è 2A3.

1.3 RAPPRESENTAZIONE INTERNA DELLE INFORMAZIONI

Le 256 configurazioni diverse cui possono dar luogo gli 8 bits significativi contenuti in un byte (escluso cioè il bit di disparità) possono essere interpretate in due modi diversi :

a) come una coppia di cifre esadecimali. Ciò corrisponde a considerare un byte come costituito da due gruppi di 4 bits (cioè da due semibytes). I quattro bits di ciascun semibyte hanno i pesi binari 1,2,4,8 e pertanto ciascuna cifra esadecimale può raggiungere il valore F,

b) come uno tra i 64 caratteri dell'alfabeto del calcolatore.

Nel primo caso tutte le 256 configurazioni sono valide e sostanzialmente il contenuto di un byte può essere interpretato come un unico valore binario compreso tra 0 e 255.

Nel secondo caso soltanto 64 delle 256 configurazioni binarie possibili sono valide e precisamente quelle delle colonne 6, 7, 8 e 9 della tabella di fig. 1.1. I 64 caratteri di questa tabella si trovano sulla tastiera della unità di input e sono anche stampabili dalla unità di output.

Fa eccezione il carattere ƀ (spazio) che non corrisponde ad alcun simbolo grafico della unità di stampa (che pertanto è in grado di stampare soltanto 63 caratteri) e viene utilizzato per creare spaziature nelle righe di stampa.

Le informazioni che possono essere registrate nella memoria del calcolatore CND possono quindi essere suddivise in due classi:

a) dati esadecimali

b) dati alfanumerici

I dati esadecimali, tra i quali sono comprese anche le istruzioni costituenti il programma, sono costituiti da uno o più cifre esadecimali contenute in un numero intero di bytes.

Semibyte di destra ES.	BIN.	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0	0000							0	ƀ	|	+						
1	0001							1	A	J	/						
2	0010							2	B	K	S						
3	0011							3	C	L	T						
4	0100							4	D	M	U						
5	0101							5	E	N	V						
6	0110							6	F	O	W						
7	0111							7	G	P	X						
8	1000							8	H	Q	Y						
9	1001							9	I	R	Z						
A	1010							[	&	-	_						
B	1011							#	.	$	'						
C	1100							@	]	*	%						
D	1101							:	(	)	=						
E	1110							>	<	;	"						
F	1111							?	\	,	!						

(Column headers: Semibyte di sinistra)

Fig. 1.1 - Alfabeto del Calcolatore CND

I dati esadecimali dovranno cioè avere la cifra meno significativa nel semi-byte di destra di un byte e dovranno, eventualmente, essere completati da uno zero nel semi-byte di sinistra del byte più significativo.

Quando un dato esadecimale deve essere sottoposto ad una operazione aritmetica, esso viene considerato sempre positivo.

I dati alfanumerici sono costituiti da uno o più caratteri dell'alfabeto del calcolatore che occupano altrettanti bytes consecutivi (ad esempio un nome di persona, la descrizione di un prodotto, un importo, un codice indicativo, costituiscono dati alfanumerici).

Tra essi sono di particolare rilievo i dati decimali che sono costituiti da una o più cifre decimali la cui codificazione binaria è riportata nelle prime dieci posizioni della colonna 6 della tabella di fig. 1.1 seguite eventualmente a destra dal carattere indicante il segno (caratteri + o - della stessa tabella).

Anche il segno, quando esiste, occupa un byte e precisamente quello che si trova a destra del byte che contiene la cifra meno significativa del dato decimale.

Soltanto i dati decimali possono essere oggetto di operazioni aritmetiche decimali, incluso il confronto algebrico, mentre qualsiasi dato alfanumerico può essere utilizzato come termine in una operazione di confronto detta di 'confronto logico' che non tiene conto del particolare significato del segno algebrico.

Sono riportati, di seguito, alcuni esempi della rappresentazione in memoria di dati di vario tipo.

a) Il dato esadecimale 4AB75F è registrato nei bytes compresi tra quello di indirizzo 3A7 (935 decimale) e quello di indirizzo 3A9 (937 decimale).

b) Il dato alfanumerico ROSSI ƀ ANTONIO è registrato nei bytes compresi tra quello di indirizzo OBF4 (3060 in decimale) e quello di indirizzo OCOO (3072 in decimale).

c) Il dato decimale -124795 è registrato nei bytes compresi tra quello di indirizzo 0E00 (3584 in decimale) e quello di indirizzo 0E06 (3590 in decimale).

1.4 L'UNITA' DI GOVERNO

Dati gli scopi del nostro corso, ci limiteremo a dare soltanto brevi cenni sull'unità di governo del calcolatore CND.

Questa unità ha il compito di svolgere le seguenti funzioni :

a) prelevare dalla memoria le istruzioni che costituiscono il programma,

b) interpretare le istruzioni, cioè determinare quale operazione deve essere eseguita e individuare gli operandi,

c) eseguire effettivamente l'operazione.

Queste funzioni vengono eseguite sequenzialmente dall'unità di governo del calcolatore CND, mentre su calcolatori di maggiori dimensioni possono essere svolte in sovrapposizione. Ciò significa ad esempio,che mentre è in corso di esecuzione la istruzione n-esima l'unità di governo può già interpretare l'istruzione (n+1)-esima.

Le funzioni dell'unità di governo vengono svolte tramite circuiti elettronici realizzati mediante diodi e transistori, componenti micrologici, ecc.

Tali circuiti elettronici sono progettati tenendo conto che tutte le operazioni aritmetiche e tutte le operazioni necessarie per il prelevamento delle istruzioni dalla memoria, la loro interpretazione e la loro esecuzione sono realizzabili mediante opportune combinazioni di pochissime operazioni elementari dette "somma logica","prodotto logico" e "negazione logica".

Tali operazioni agiscono su variabili binarie che sono rappresentate dai due valori di tensione che possono presentarsi agli ingressi e all'uscita di ogni circuito elettronico presente nell'unità di governo.

Le operazioni elementari di cui sopra vengono spesso indicate con

i nomi "OR","AND", e "NOT" e sono realizzate mediante semplici circuiti che vengono usualmente indicati con i simboli grafici della fig. 1.2.

Se indichiamo con "zero" e "uno" i valori assunti dalla variabile binaria su cui agiscono le operazioni OR, AND e NOT (usualmente si associa "zero" al valore nullo della tensione, e "uno" al valore positivo) il funzionamento di tali operazioni può essere definito dalla tabella di fig. 1.3.

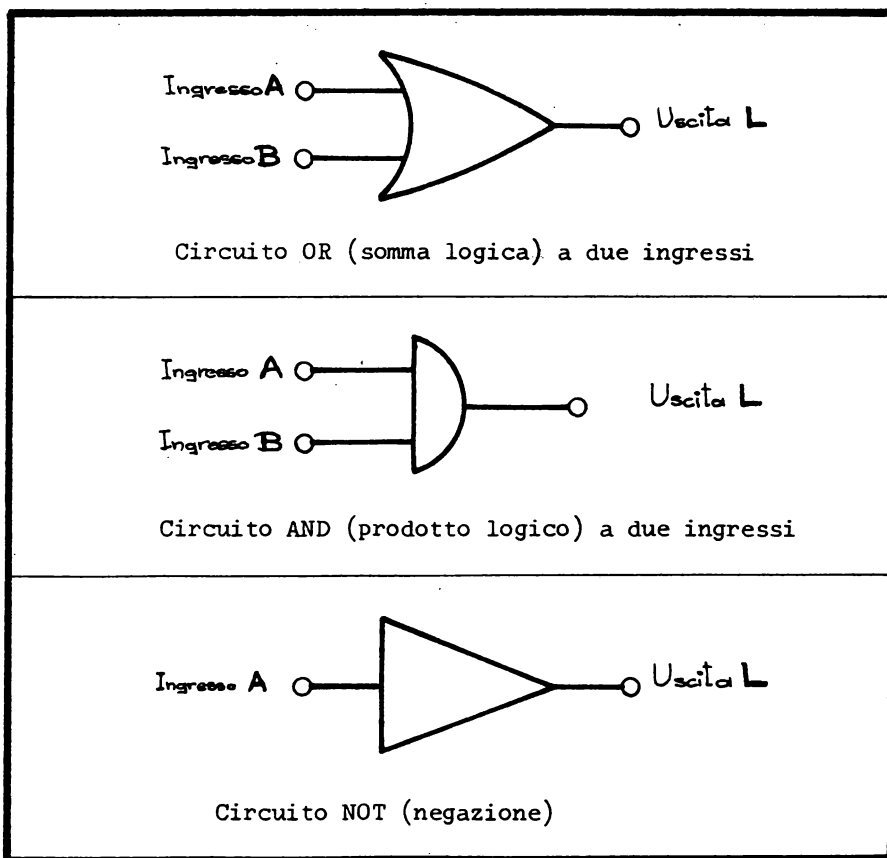



Fig. 1.2 - Circuiti logici elementari

Nel calcolatore CND i circuiti elettronici elementari di cui si è parlato sono impiegati per realizzare un insieme di istruzioni elementari dette micro-istruzioni.

L'insieme di istruzioni che costituiscono il vero e proprio linguaggio macchina del calcolatore è realizzato mediante sequenze di micro-istruzioni registrate in modo permanente in una memoria non accessibile all'utilizzatore e avente elevate caratteristiche di velocità detta memoria "READ ONLY".

Tale memoria può essere oggetto di lettura, ma non di scrittura e il suo contenuto non può pertanto essere modificato.

Essa viene brevemente indicata con il termine ROM (read only memory) e può contenere, oltre alle sequenze di micro-istruzioni che realizzano le istruzioni di macchina, anche dei veri e propri programmi, quali ad esempio i programmi di caricamento e di lancio che, come si vedrà in seguito, provvedono all'avviamento delle elaborazioni.

ingresso A	0	0	1	1
ingresso B	0	1	0	1
uscita A "OR" B	0	1	1	1
uscita A "AND" B	0	0	0	1
uscita "NOT" A	1	1	0	0

Fig.1.3 - Tabella di funzionamento dei circuiti elementari

1.5 IL QUADRO DI COMANDO

Il quadro di comando (console) del calcolatore CND contiene una tastiera esadecimale utilizzabile per l'introduzione di informazioni esadecimali nel sistema, e un certo numero di tasti e di indicatori luminosi le cui funzioni saranno illustrate nel seguito.

1.5.1 Tastiera esadecimale

La tastiera esadecimale, come risulta dalla rappresentazione schematica della console riportata nella figura 1.4 comprende 16 tasti corrispondenti alle cifre esadecimali da 0 ad F.

Normalmente la tastiera esadecimale è bloccata per cui l'operatore non può premere alcun tasto. E' compito dei programmi registrati nella memoria "read only" dell'unità di governo o dei programmi dell'utilizzatore registrati nella memoria a nuclei comandare lo sblocco della tastiera tramite le opportune istruzioni di input.

Come si vedrà meglio nel seguito, l'operatore viene avvisato della necessità di premere uno dei tasti della tastiera esadecimale dall'accendersi di una lampada della console. Inoltre il valore della cifra introdotta viene visualizzato su opportuni lampadini della console stessa.

1.5.2 Tasti ON, OFF e lampada ON

La pressione del tasto ON serve a fornire tensione all'unità centrale del calcolatore cioè a governo, memoria e console, mentre il tasto OFF viene utilizzato per togliere la tensione.

Quando l'unità centrale è sotto tensione la lampada ON che si

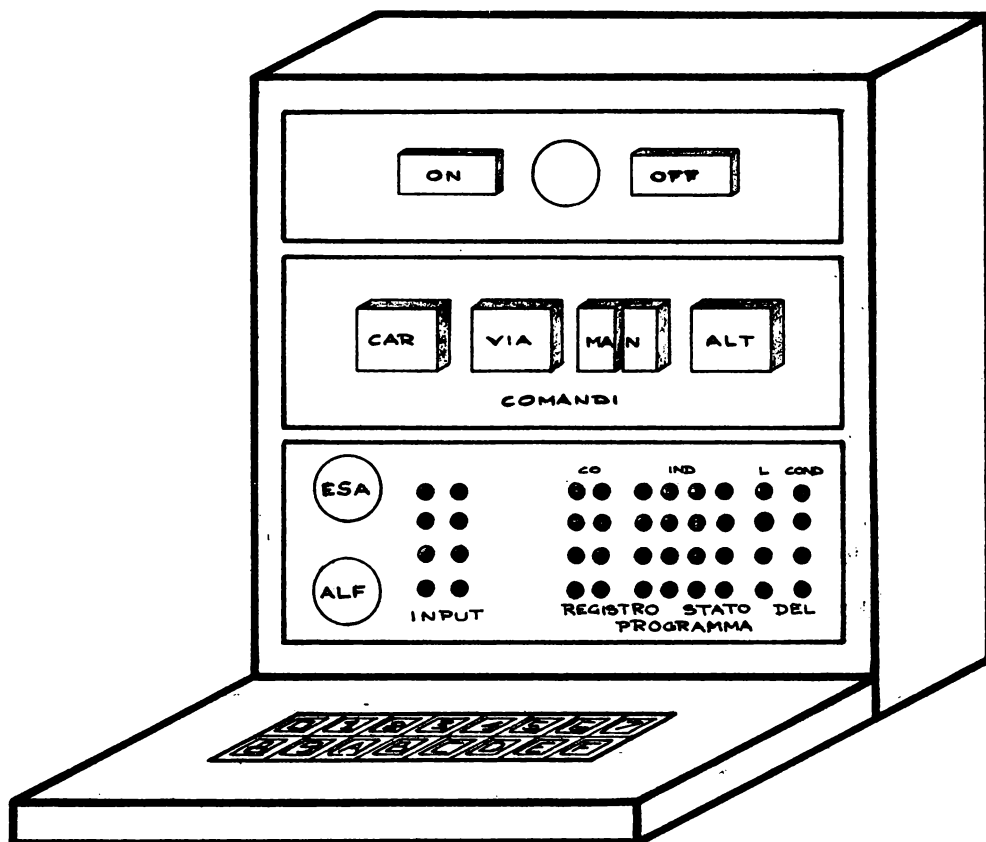



Fig. 1.4 - Il quadro di comando (CONSOLE)

trova tra due tasti ora descritti è accesa.

La tensione alla unità di input/output viene data o tolta in modo autonomo tramite opportuni tasti che si trovano sull'unità stessa.

1.5.3 Il Tasto CAR

Questo tasto viene premuto ogni volta che si vuole "caricare", cioè registrare nella memoria, un programma che dovrà essere successivamente eseguito. La sua pressione avvia l'esecuzione di un programma, detto "Programma di caricamento" e registrato in modo permanente nella ROM il quale svolge le seguenti funzioni

a) accende la lampada ESA per segnalare all'operatore che la macchina è in stato di attesa di dati esadecimali,

b) sblocca la tastiera esadecimale consentendo l'introduzione di un primo gruppo di quattro cifre esadecimali che vengono considerate come indirizzo di caricamento,

c) registra l'indirizzo di caricamento in una zona della memoria riservata alla unità di governo,

d) consente l'introduzione dalla tastiera esadecimale delle cifre esadecimali che compongono il programma e le registra, in ragione di 2 per ogni byte, nella memoria a partire dall'indirizzo di caricamento.

Quando l'operatore desidera interrompere il caricamento, perchè tutto il programma è stato caricato o perchè si è reso conto di avere commesso qualche errore di battitura, deve premere il tasto ALT.
Una nuova pressione del tasto CAR e l'introduzione di un diverso indirizzo di caricamento permettono all'operatore la correzione degli eventuali errori.

Si osservi che, poichè la memoria del calcolatore CND ha come indirizzo massimo il valore esadecimale FFF (4095 esadecimale) la prima cifra dell'indirizzo di caricamento deve essere necessariamente zero. Essa deve tuttavia essere battuta per permettere il caricamento di due bytes completi.

1.5.4 Il Tasto VIA

La funzione di questo tasto è di dare il via alla esecuzione di un programma precedentemente caricato in memoria.

La sua pressione avvia l'esecuzione di un programma, detto "Programma di lancio", registrato in modo permanente nella ROM, il quale svolge le seguenti funzioni :

a) accende la lampada ESA per segnalare all'operatore che la macchina è in stato di attesa di dati esadecimali,

b) sblocca la tastiera esadecimale di console consentendo l'introduzione di quattro cifre esadecimali (la prima delle quali dovrà essere zero),

c) memorizza l'indirizzo costituito dalle cifre così introdtte in una zona riservata della memoria,

d) lancia l'esecuzione del programma registrato in memoria partendo dall'istruzione che si trova all'indirizzo introdotto dall'operatore.

Osserviamo che l'indirizzo di lancio non coincide necessariamente con l'indirizzo di caricamento del programma registrato in memoria; infatti il programmatore potrebbe ad esempio avere posto in testa al suo programma delle costanti da utilizzare nel corso del programma stesso, oppure nella memoria potrebbero essere stati registrati più programmi il cui ordine di esecuzione è diverso da quello di introduzione.

L'esecuzione del programma normalmente continua fino a che viene incontrata una istruzione ALT (vedi par. 3.3), ma può essere interrotta dall'operatore mediante la pressione del tasto MAN o del tasto ALT.

1.5.5 Il Tasto MAN

Questo tasto è composto di due parti. La parte di sinistra è bi-stabile cioè può assumere due posizioni distinte.

Quando questa parte del tasto viene abbassata, l'esecuzione del programma viene arrestata dopo che è stata completata l'istruzione in corso e il calcolatore si pone in uno stato che chiameremo "manuale".

In tale stato la parte di destra del tasto MAN viene abilitata al funzionamento ed il calcolatore esegue una istruzione del programma in corso ad ogni pressione di tale parte del tasto.

Si dice che il programma viene così eseguito "passo a passo".

Questo modo di funzionamento permette la lettura dei lampadini che mettono in evidenza il contenuto del cosidetto "registro stato del programma" illustrato nel seguito (paragrafo 2.2).

Quando la parte di sinistra del tasto MAN viene riportata nella posizione normale, il programma in corso riprende la sua esecuzione automatica e la parte destra del tasto è disabilitata.

1.5.6 Il Tasto ALT

La pressione del tasto ALT durante l'esecuzione di un qualsiasi programma determina l'arresto di tale esecuzione senza possibilità di ripresa.

Più in particolare la pressione del tasto ALT provoca quanto segue :

a) pone a zero il contenuto del registro stato del programma,

b) spegne le lampade ESA ed ALF eventualmente accese,

c) blocca le tastiere esadecimale e alfanumerica se queste sono sbloccate,

d) abilita l'uso dei tasti CAR e VIA che sono inoperanti in fase di esecuzione di un programma.

Il tasto ALT è efficace anche durante l'esecuzione dei programmi di caricamento e di lancio.

1.5.7 Le lampade ESA e ALF

Come si è già detto la lampada ESA viene accesa dai programmi di caricamento e di lancio ogni volta che è richiesta l'introduzione di cifre esadecimali.

Poichè anche un programma scritto dall'utilizzatore può chiedere l'introduzione di cifre esadecimali, la lampada ESA viene accesa anche dall'istruzione di input esadecimale IES (paragrafo 3.2).

La lampada ALF viene accesa dalle istruzioni di input alfanumerico IAL (paragrafo 3.2) che si trovano nel programma dell'utilizzatore ed ha la funzione di avvisare l'operatore che il calcolatore è in attesa dell'introduzione di caratteri alfanumerici.

Contemporaneamente alla accensione di una qualsiasi delle due lampade si ha lo sblocco della relativa tastiera; quando la introduzione dei dati è terminata la lampada accesa viene spenta e la tastiera nuovamente bloccata.

Le lampade ESA e ALF sono affiancate da otto lampadini sui quali viene visualizzato il contenuto binario del byte che è stato interessato dall'ultima introduzione effettuata.

Ovviamente i quattro lampadini di sinistra si riferiscono al semi-byte più significativo, mentre i lampadini di destra si riferiscono al semi-byte meno significativo.

1.5.8 I lampadini del registro stato del programma

Questi lampadini permettono la visualizzazione del contenuto di una particolare zona riservata di memoria, detta "registro stato del programma", nella quale sono immagazzinate alcune informazioni essenziali sullo stato del programma in corso di esecuzione.

La lettura di tali informazioni è possibile soltanto quando la parte sinistra del tasto MAN è abbassata e l'esecuzione del programma procede "passo a passo"; durante l'esecuzione automatica del programma l'operatore potrà invece vedere soltanto una rapidissima successione di accensioni e spegnimenti che non gli forniscono informazioni quantitative.

I lampadini sono tanti quanti sono i bits che costituiscono il registro stato del programma e cioè 32 e sono disposti in 8 colonne di quattro lampadini ciascuno.

1.6 LA UNITA' DI INPUT-OUTPUT

Come si è già detto l'unità di input-output del calcolatore CND è costituita da una telescrivente schematizzata nella fig. 1.5.

La tastiera, rappresentata più in dettaglio nella fig. 1.6, comprende i tasti che permettono di introdurre i 64 caratteri dell'alfabeto del calcolatore e un certo numero di tasti di servizio.

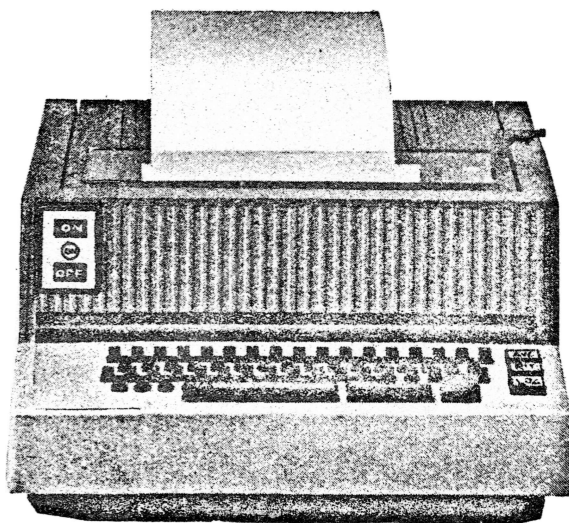



Fig. 1.5 - L'unità di input-output

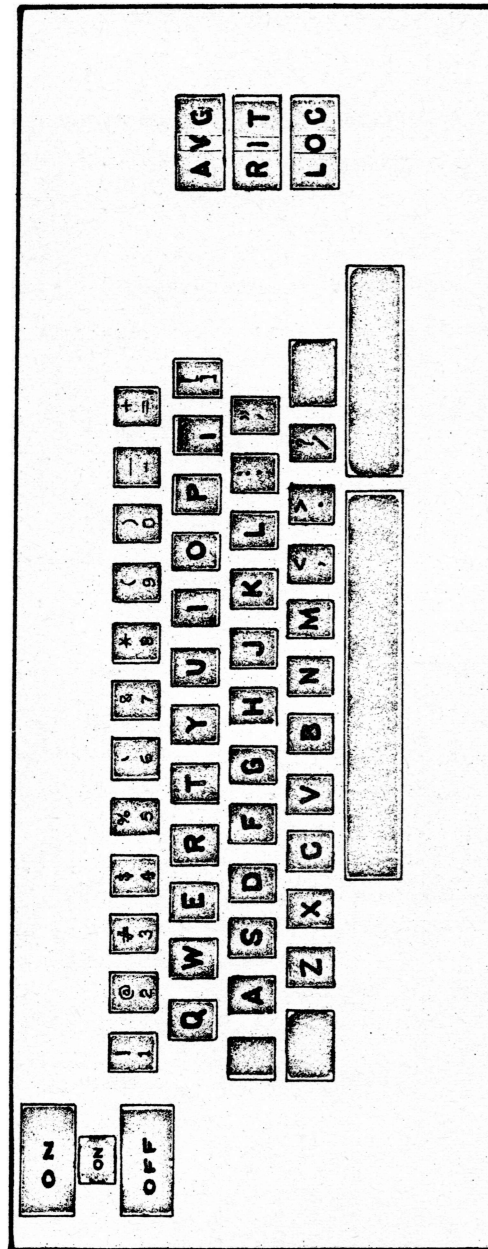



Fig. 1.6 - La tastiera di input

In particolare sono disponibili :

- i tasti ON, OFF per dare e togliere tensione all'unità,
- la lampada ON accesa quando l'unità è sotto tensione,
- il tasto bistabile LOC (locale) che permette di scollegare la telescrivente dal calcolatore CND e di usarla come una semplice macchina per scrivere "off-line",
- il tasto RIT per il ritorno a capo manuale del carrello,
- il tasto AVC che permette di far avanzare manualmente la carta,
- una barra per la tabulazione.La pressione della barra provoca l'introduzione del carattere ƀ (spazio) e l'avanzamento di una posizione del carrello,
- un dispositivo, equivalente a quello per le lettere maiuscole di una comune macchina per scrivere, per permettere di introdurre e stampare l'uno o l'altro dei due caratteri riportati su un certo numero di tasti (ricordiamo che l'alfabeto non comprende lettere minuscole).

Il dispositivo per la stampa permette l'uso di carta a striscia continua o a soffietto ed il carrello è in grado di accettare la stampa di righe comprendenti un massimo di 100 caratteri.

Il ritorno a capo del carrello è automatico insieme con l'interlinea ed avviene prima dell'esecuzione di ogni istruzione di input e di output comandata da programma interessanti l'unità.

Per ottenere interlinee supplementari si ricorre alla stampa di una riga contenente soltanto spazi (caratteri ƀ).

L'uso opportuno del carattere ƀ permette anche di ottenere tabulati organizzati in modo razionale ed esteticamente valido.

Quando la stampa è comandata dal calcolatore i caratteri figurano in nero, mentre quando è provocata manualmente dall'operatore che preme la tastiera viene fatta in colore rosso per mettere in risalto la diversa provenienza.

Va osservato che quando l'unità è collegata "on-line" l'operato re può premere la tastiera soltanto quando il programma in fase di esecuzione incontra una istruzione di input alfanumerico che come già detto sblocca la tastiera e provoca anche l'accensione della lampada ALF di console.

In ogni altro momento la tastiera è bloccata e quindi non uti - lizzabile.

1.7 SEQUENZE OPERATIVE PER L'UTILIZZAZIONE DEL CALCOLATORE

Da quanto è stato detto nei paragrafi precedenti possono essere dedotte le operazioni che devono essere effettuate sulla console. Per maggior chiarezza esse saranno tuttavia riassunte schematicamente nel seguito.

Si suppone che la tensione all'intero sistema (unità centrale più unità di input / output) sia stata fornita mediante la pressione dei tasti ON che si trovano sulla console e sull'unità di input-output.

1.7.1 Introduzione di un programma

Per effettuare l'introduzione di un programma si procede alle seguenti operazioni :

a) pressione del tasto ALT per porre a zero il registro stato del programma e per abilitare l'uso del tasto CAR.

b) Pressione del tasto CAR.

c) Battitura sulla tastiera esadecimale di 4 cifre esadecimali (la prima delle quali zero) che rappresentano l'indirizzo di caricamento del programma.

d) Battitura sulla tastiera esadecimale delle cifre che costituiscono il programma.

e) Pressione del tasto ALT.

La figura 1.7 illustra schematicamente la procedura ora descritta e le funzioni svolte dal programma di caricamento. Le operazioni svolte dall'operatore sono indicate mediante rettangoli tratteggiati.

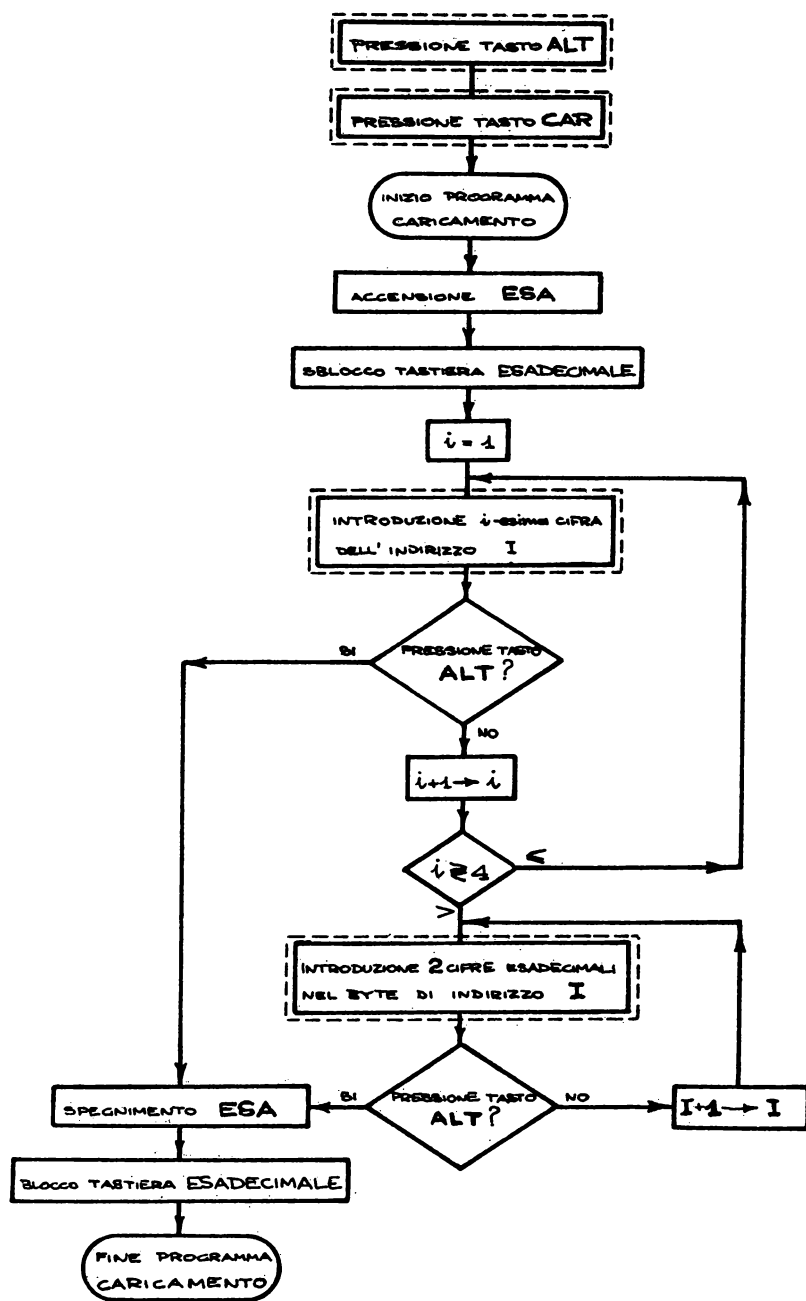



Fig. 1.7 - Caricamento di un programma

Si osservi che la pressione del tasto ALT in qualunque momento (per esempio durante l'introduzione dell'indirizzo) provoca la interruzione del programma di caricamento che può essere fatta ripartire da capo mediante la pressione del tasto CAR.

1.7.2 Lancio di un programma

Per effettuare il lancio di un programma si procede alle seguenti operazioni :

a) Pressione del tasto ALT (operazione necessaria soltanto se dopo l'ultima volta in cui il tasto ALT è stato premuto si è avuta la pressione del tasto CAR o del tasto VIA).

b) Pressione del tasto VIA.

c) Battitura sulla tastiera esadecimale di 4 cifre esadecimali (la prima delle quali zero) che rappresenta l'indirizzo della istruzione di programma che deve essere eseguita per prima (indirizzo di lancio).

La figura 1.8 illustra schematicamente la procedura ora descritta e le funzioni svolte dal programma di lancio. Le operazioni svolte dall'operatore sono indicate mediante rettangoli tratteggiati.

Si osservi che la pressione del tasto ALT in qualunque momento, e cioè sia durante l'introduzione dell'indirizzo sia durante la esecuzione del programma dell'utente,provoca l'arresto della elaborazione.

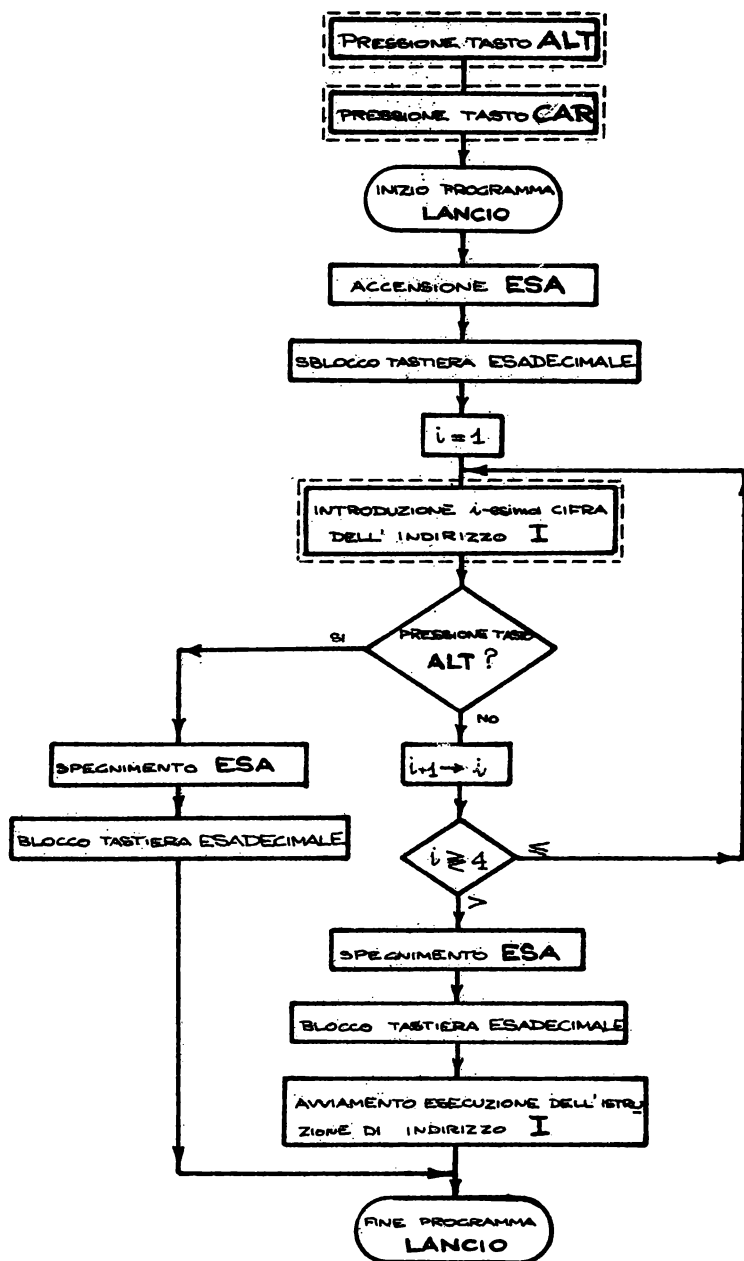



Fig. 1.8 - Lancio di un programma

1.8 LA ZONA RISERVATA DELLA MEMORIA

E' stato detto in precedenza che la memoria del calcolatore CND ha una capacità di 4096 bytes e che gli indirizzi di memoria variano da 0 a FFF esadecimale.

Non tutti questi bytes sono però disponibili liberamente per lo utilizzatore, in quanto i primi 64 (cioè quelli di indirizzo compreso tra zero e 3 F esadecimale incluso) sono riservati alla unità di governo e costituiscono la cosidetta "zona riservata della memoria".

Ne consegue che i programmi e i dati dell'utilizzatore dovranno avere un indirizzo superiore o uguale a 40 esadecimale. All'utilizzatore è permesso prelevare informazioni dalla zona riservata di memoria, mentre non è possibile registrare delle informazioni in tale zona.

I 64 bytes componenti la zona riservata della memoria vengono utilizzati per contenere le seguenti informazioni :

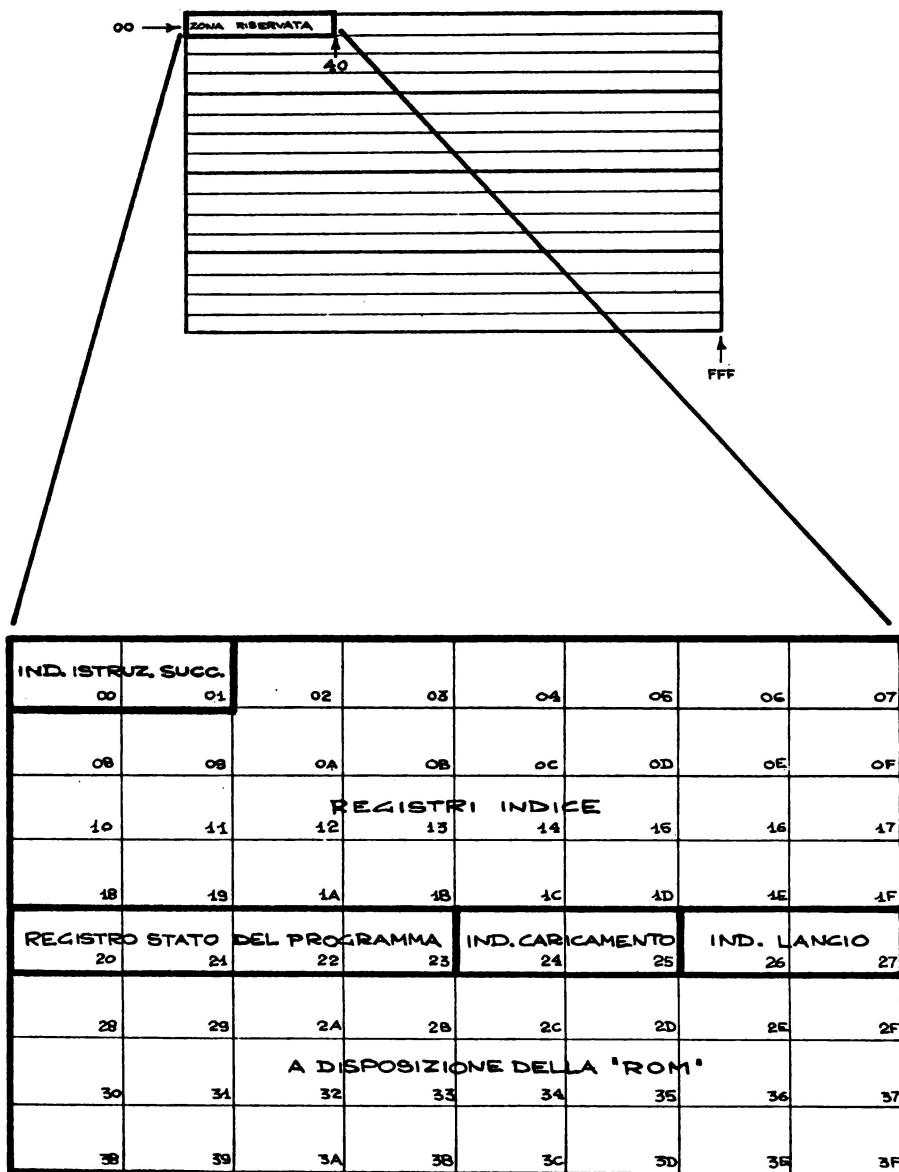
- bytes 00 ÷ 01 In questi due bytes viene memorizzato, a cura dell'istruzione SME (paragrafo 3.3), l'indirizzo dell' istruzione ad essa successiva.
- bytes 02 ÷ 1F Questi bytes non sono utilizzati dal calcolatore CND nella sua versione base. Trovano impiego in una versione più estesa come registri indice (si veda il CAPITOLO 6).
- bytes 20 ÷ 23 Sono registrate in questi quattro bytes, che costituiscono il cosidetto "registro stato del programma", alcune informazioni essenziali sullo stato del programma stesso. Tali informazioni verranno descritte in dettaglio nel paragrafo seguente.
- bytes 24 ÷ 25 Viene memorizzato in questi due bytes l'indirizzo esadecimale introdotto dall'operatore durante l'esecuzione del programma di caricamento.

- bytes 26 ÷ 27. Viene memorizzato in questi due bytes l'indirizzo esadecimale introdotto dall'operatore durante l'esecuzione del programma di lancio (indirizzo di lancio).

- bytes 28÷3F. Questi bytes vengono utilizzati dai microprogrammi memorizzati nella ROM come zone transitorie di deposito (ad esempio per le operazioni aritmetiche).

Nella figura 1.9 è schematizzata la mappa della zona riservata dalla memoria.

Fig. 1.9 - Mappa della zona riservata della memoria

1.9 IL REGISTRO STATO DEL PROGRAMMA

Come si è detto in precedenza, alcune essenziali informazioni sullo stato del programma in corso di elaborazione sono immagazzinate e continuamente aggiornate nei bytes 20÷23 della zona riservata della memoria; tali bytes costituiscono il cosiddetto "registro stato del programma" (program status word).

Il contenuto del registro stato del programma è visualizzato sulla console mediante 32 lampadini, corrispondenti ai 32 bits del registro. Una mappa dettagliata del registro è riportata nella figura 1.10.

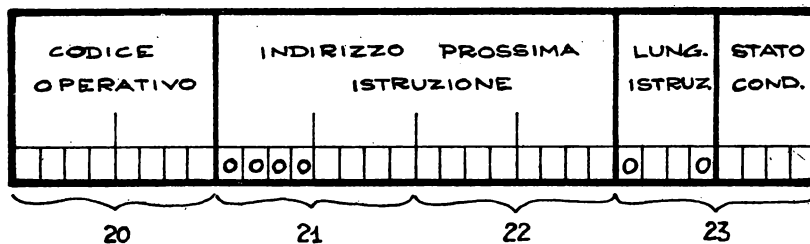



Fig.1.10 - Mappa del "Registro Stato del Programma"

Le informazioni contenute nel registro stato del programma sono le seguenti :

- byte 20 Questo byte contiene il codice operativo (paragrafo 2.1) della istruzione che è in corso di esecuzione.

- bytes 21, 22 Questi due bytes contengono il valore esadecimale dell'indirizzo della istruzione che dovrà essere eseguita dopo l'istruzione in corso. Tale indirizzo viene modificato alla fine della fase di interpretazione di ogni istruzione

e ad esso viene aggiunta la lunghezza dell'istruzione interpretata.
Esso può essere modificato dalle istruzioni di salto il cui effetto è di inserire nel registro stato del programma l'indirizzo della istruzione di arrivo del salto stesso (paragrafo 3.3).
L'unità di governo del calcolatore si serve dell'indirizzo memorizzato nel registro stato del programma per prelevare l'istruzione successiva.
Ovviamente i quattro bits più significativi di tale indirizzo sono sempre zero nel calcolatore CND il cui indirizzo massimo di memoria è FFF.

- byte 23 (semibyte di sinistra) Questa zona del registro stato del programma contiene la lunghezza dell'istruzione in corso di esecuzione. Come si vedrà più avanti le istruzioni del calcolatore CND possono avere lunghezze di 4 e 6 bytes, quindi il primo e l'ultimo bit della zona sono sempre nulli.
  La lunghezza dell'istruzione viene determinata dal calcolatore in base al codice operativo (paragrafo 2.1).

- byte 23 (semibyte di destra) Questi quattro bits memorizzano lo stato delle condizioni interne provocato dall'ultima operazione aritmetica o di confronto eseguita e la presenza di condizioni di errore.
  In particolare il significato dei due bits di destra è il seguente :

  valore 00 : l'ultimo confronto ha dato risultato uguale oppure l'ultima operazione aritmetica ha dato risultato nullo.

  valore 01 : l'ultimo confronto ha dato risultato minore oppure l'ultima operazione aritmetica ha dato risultato negativo.

  valore 10 : l'ultimo confronto ha dato risultato maggiore oppure l'ultima operazione aritmetica ha dato risultato positivo.

valore 11 : l'ultima operazione aritmetica ha provocato "over flow" (traboccamento). Questo può verificarsi ad esempio se il minuendo di una sottrazione è minore del sottraendo (si veda il paragrafo 3.1).

Il terzo bit assume il valore 1 quando l'istruzione in corso non può essere eseguita dal calcolatore. In questo caso ha luogo la condizione di "errore programma" ("program check") e l'elaborazione è arrestata definitivamente (paragrafo 5).

Il quarto bit non è utilizzato nella versione base del calcolatore CND.

CAPITOLO 2

IL LINGUAGGIO MACCHINA

2.1 GENERALITA'

2.2 FORMATI DELLE ISTRUZIONI

2.3 INTERPRETAZIONE ED ESECUZIONE DELLE ISTRUZIONI

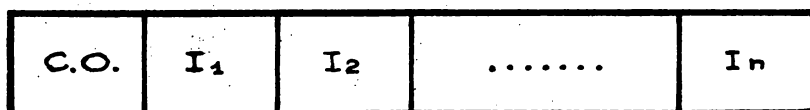
2.4 REDAZIONE DI UN PROGRAMMA

2.1 GENERALITA'

Come è noto un programma redatto nel linguaggio macchina di un calcolatore è costituito da una sequenza di istruzioni registrate nella memoria principale.

Ogni istruzione può essere considerata logicamente suddivisa in due parti; la prima di queste caratterizza il tipo di operazione che deve essere eseguita, la seconda individua gli operandi interessati nella memoria del calcolatore.

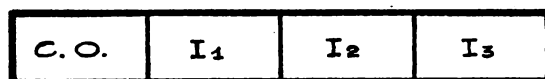
Il tipo di operazione viene specificato con un codice che può assumere tanti valori quante sono le istruzioni che fanno parte del "repertorio" di istruzioni del calcolatore. Tale codice viene chiamato "codice operativo" o "codice di funzione".

Ciascuno degli operandi è individuato da un indirizzo e pertanto ogni istruzione avrà un formato del tipo seguente :

dove C.O. è il codice operativo, I_1, I_2, ..., I_n sono gli indirizzi rispettivamente del primo, del secondo, dell'n-esimo operando.

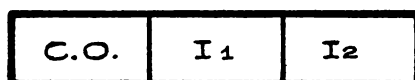
Nei casi concreti gli indirizzi specificati non sono mai più di tre e spesso sono soltanto due o addirittura uno solo.

Il fatto che gli indirizzi specificati siano meno di tre non sta a significare che anche gli operandi siano effettivamente in numero minore di tre, cosa che potrebbe non avere significato per le operazioni aritmetiche elementari, ma che gli indirizzi di alcuni di essi sono impliciti.

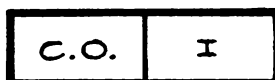
Così ad esempio una istruzione di sottrazione può avere in tre diversi calcolatori uno dei seguenti formati :

a)

dove I_1, I_2, I_3 sono rispettivamente gli indirizzi del minuendo del sottraendo e della differenza

b)

C.O.	I1	I2

dove I_1, e I_2 sono gli indirizzi del minuendo e del sottraendo, mentre la differenza viene costruita al posto del minuendo

c)

C.O.	I

dove I è l'indirizzo del sottraendo. In tal caso si suppone che il minuendo si trovi in una posizione convenzionale detta "accumulatore" il cui indirizzo è sottointeso.

Anche la differenza viene in questo caso costruita nell'accumulatore.

Come si vedrà nel prossimo paragrafo le istruzioni aritmetiche del calcolatore CND sono del tipo b) cioè a due indirizzi.

Il "repertorio" delle istruzioni è ovviamente variabile da un calcolatore all'altro; è tuttavia possibile individuare le seguenti categorie di istruzioni che sono sicuramente presenti in tutti i calcolatori :

- istruzioni aritmetiche o di confronto
- istruzioni di input-output
- istruzioni di controllo

Tra le istruzioni aritmetiche si possono far rientrare anche le istruzioni di trasferimento la cui funzione è di trasferire dati da una zona all'altra della memoria.

Per quanto riguarda le istruzioni aritmetiche vere e proprie è opportuno osservare che calcolatori di piccole dimensioni, e quindi anche il calcolatore CND, sono spesso dotati delle sole istruzioni di addizione e sottrazione.

In questo caso la moltiplicazione e la divisione sono considerate algoritmi non elementari e vengono programmate ad esempio come sequenze rispettivamente di addizione e di sottrazione.

Le istruzioni di confronto permettono di stabilire, dati due operandi A e B, quale delle tre relazioni

$$A > B \qquad A = B \qquad A < B$$

è verificata. Il risultato del confronto viene memorizzato mediante opportuni codici in particolari posizioni di memoria che nel calcolatore CND fanno parte del "registro stato del programma".

Le istruzioni di input/output permettono il trasferimento di dati da una unità periferica o da una eventuale memoria di massa alla memoria principale e viceversa.

La struttura di queste istruzioni è particolarmente semplice nel caso del calcolatore CND che è dotato di unità di input/output piuttosto elementari.

Lo scambio di informazioni tra memoria principale e unità periferiche è però in generale una delle funzioni più complesse in quanto è necessario tener presente una serie di problemi, quali lo sfruttamento delle possibilità di sovrapposizione, la verifica della disponibilità delle unità, il controllo dell'esattezza della operazione di trasferimento e così via.

Le istruzioni di controllo sono essenzialmente ordini di salto che permettono di alterare l'ordine di esecuzione delle istruzioni componenti il programma eventualmente in modo subordinato al verificarsi di una determinata condizione che può essere il risultato di un confronto.

2.2 FORMATI DELLE ISTRUZIONI

Il linguaggio macchina del calcolatore CND è composto da istru - zioni i cui formati possono essere raggruppati in due categorie: formato A e formato B.

La lunghezza delle istruzioni di formato A è di 6 bytes (48 bits 12 cifre esadecimali), mentre le istruzioni di formato B sono lunghe 4 bytes (32 bits, 8 cifre esadecimali).

Esaminiamo ora in dettaglio i due formati.

2.2.1 Le istruzioni di formato A

Le istruzioni di formato A hanno il seguente tracciato :

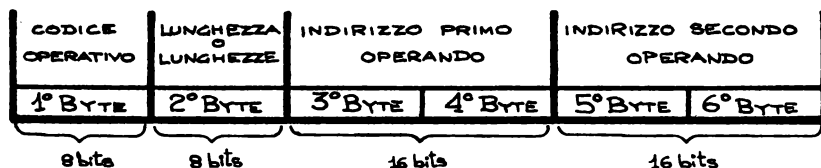



Fig. 2.1 - Formato A

Tutte le istruzioni di questo formato operano sul contenuto di due campi di memoria detti rispettivamente "1° operando" e " 2° operando".

Nel 1° byte (8 bits, 2 cifre esadecimali) compare il "codice operativo", che serve per distinguere in modo biunivoco le varie istruzioni. In tutte le istruzioni di formato A la prima cifra

esadecimale del"codice operativo" è A.

Nel 2° byte (8 bits, 2 cifre esadecimali) compaiono due valori esprimenti le lunghezze dei due operandi, oppure, a seconda delle istruzioni, un unico valore esprimente la lunghezza di un dato. I valori delle lunghezze che compaiono nel 2° byte delle istruzioni sono i valori reali diminuiti di 1 unità. Così, ad esempio, per indicare che un dato è lungo 3 bytes si dovrà scrivere 2 nella posizione corrispondente alla sua lunghezza, e per indicare che un dato è lungo 1 byte si dovrà scrivere 0.

Questo accorgimento è usato anche per le istruzioni di formato B

I valori massimi che possono essere indicati nel 2° byte sono 15 o 255 a seconda delle istruzioni e corrispondono alle lunghezze effettive 16 e 256.

Nel 3° e 4° byte (16 bits, 4 cifre esadecimali) compare il valore dell'indirizzo di memoria di un dato detto 1° operando.

Il valore di tale indirizzo viene espresso utilizzando la numerazione esadecimale.

Poichè il calcolatore CND ha nella sua versione base una memoria con 4096 posizioni, il massimo valore che può comparire come indirizzo è in esadecimale FFF; pertanto la prima cifra esadecimale dell'indirizzo viene riempita sempre con 0 (zero).

Il 5° ed il 6° byte (16 bits, 4 cifre esadecimali) svolgono funzioni analoghe a quelle dei due bytes precedenti contenendo l'indirizzo di memoria di un altro dato, detto 2° operando.

Ad ogni istruzione viene assegnato anche un "codice simbolico", composto da tre caratteri alfabetici, che non compare nel formato ora descritto e che ha una funzione esclusivamente mnemonica.

Il linguaggio macchina del calcolatore CND comprende le seguenti istruzioni di formato A :

ISTRUZIONI	CODICE SIMBOLICO	CODICE OPERATIVO
Addizione Decimale	ADE	A1
Sottrazione Decimale	SDE	A2
Addizione Esadecimale	AES	A3
Sottrazione Esadecimale	SES	A4
Trasferimento	TRA	A5
Confronto Logico	CLO	A6
Confronto Algebrico	CAL	A7

Fig. 2.2 - Istruzioni di Formato A

2.2.2 Le istruzioni di formato B

Le istruzioni di formato B hanno il seguente tracciato

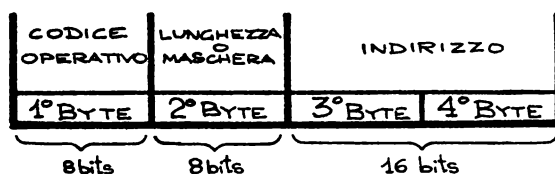



Fig. 2.3 - Formato B.

e possono essere utilizzate per introdurre od estrarre dati dalla memoria del calcolatore, oppure per controllare la sequenza logica di esecuzione delle istruzioni componenti un programma.Es

se necessitano sempre dell'indirizzo di un solo campo di memoria detto operando.

Nel 1° byte (8 bits, 2 cifre esadecimali) compare il "codice operativo" dell'istruzione.

In tutte le istruzioni di formato B la prima cifra esadecimale del "codice operativo" è "B".

Nel 2° byte (8 bits, 2 cifre esadecimali) compare un valore esprimente una lunghezza oppure, nel caso delle istruzioni di controllo, un valore di 4 bits esprimente una determinata "maschera".

Occorre tenere presente che, come nel formato A, il valore esprimente la lunghezza che compare nell'istruzione è quello effettivo diminuito di 1 unità.

La "maschera" verrà descritta in dettaglio nel paragrafo 3.3.

Nel 3° e 4° byte (16 bits, 4 cifre esadecimali) compare il valore di un indirizzo di memoria che per il motivo già esposto in precedenza avrà sempre zero come prima cifra esadecimale.

Il linguaggio macchina del calcolatore CND comprende le seguenti istruzioni di formato B

ISTRUZIONI	CODICE SIMBOLICO	CODICE OPERATIVO
Introduzione esadecimale	IES	B1
Introduzione alfanumerica	IAL	B2
Estrazione esadecimale	OES	B3
Estrazione alfanumerica	OAL	B4
Salto	SAL	B5
Salto con memorizzazione	SME	B6
Arresto	ALT	B7

Fig. 2.4 - Istruzioni di Formato B

L'istruzione di arresto ALT, pur essendo di formato B, non contiene alcuna informazione significativa oltre al codice operativo.

2.2.3 Classificazione delle istruzioni

Le istruzioni del linguaggio macchina possono essere suddivise in 3 classi in base alle funzioni da loro svolte, e precisamente :

Classe	Istruzione	Codice
ISTRUZIONI ARITMETICHE E DI CONFRONTO	Addizione decimale	(ADE)
	Sottrazione decimale	(SDE)
	Addizione esadecimale	(AES)
	Sottrazione esadecimale	(SES)
	Trasferimento	(TRA)
	Confronto logico	(CLO)
	Confronto algebrico	(CAL)
ISTRUZIONI DI INPUT/OUTPUT	Introduzione esadecimale	(IES)
	Introduzione alfanumerica	(IAL)
	Estrazione esadecimale	(OES)
	Estrazione alfanumerica	(OAL)
ISTRUZIONI DI CONTROLLO	Salto	(SAL)
	Salto con memorizzazione	(SME)
	Arresto	(ALT)

2.3 INTERPRETAZIONE ED ESECUZIONE DELLE ISTRUZIONI

Tutte le istruzioni di un programma vengono registrate nella memoria del calcolatore CND sequenzialmente per indirizzi crescenti, a partire da un indirizzo detto di caricamento registrato in precedenza nella "zona riservata di memoria" (paragrafo 1.8).

Le istruzioni vengono eseguite in modo sequenziale, salvo casi particolari (paragrafo 3.3), a partire da quella che si trova ad un indirizzo detto di lancio che viene specificato al momento della richiesta di esecuzione del programma e registrato nella "zona riservata di memoria" (paragrafo 1.8).

L'esecuzione di una istruzione da parte del calcolatore comporta lo svolgimento di una serie di ordini elementari che realizzano sui dati, i cui indirizzi sono specificati nell'istruzione stessa, le operazioni richieste (somma, sottrazione, confronto, trasferimento, lettura o stampa, ecc.).

E' chiaro che prima di potere eseguire una istruzione il calcolatore deve averla precedentemente esaminata e riconosciuta senza ambiguità.

Per poter effettuare il "riconoscimento", cioè l'"interpretazione" di una istruzione esiste nella Read Only Memory del calcolatore una "Tabella" nella quale sono riportati i "codici operativi" di tutte le istruzioni disponibili.

Se il "codice operativo" esaminato è presenta in tale "tabella", il calcolatore è in grado di riconoscere l'istruzione considerata e quindi di eseguirla.

In caso contrario l'istruzione non può essere eseguita e il programma viene arrestato.

La sequenza di micro-istruzioni che effettua il riconoscimento delle istruzioni è contenuta nella memoria "Read Only" del calcolatore, e, per le funzioni da esso svolte, si dice che realizza la "fase di interpretazione" di una istruzione.

Al termine della "fase di interpretazione" di una istruzione, se tutto è corretto, inizia la fase di esecuzione ossia viene avviata la sequenza di microistruzioni che esegue le operazioni richieste dalla istruzione stessa.

Le sequenze associate alle varie istruzioni vengono chiamate "programmi esecutivi di istruzione" e sono tutte contenute nella memoria "Read Only".

La fine della elaborazione di un "programma esecutivo di istruzione" coincide con la fine della "fase di esecuzione" di una istruzione.

Sia la sequenza che realizza la "fase di interpretazione" che le sequenze che realizzano le varie "fasi di esecuzione" vengono registrate in modo permanente nella "Read Only Memory" al momento della costruzione del calcolatore.

Un flow-chart di massima della successione delle fasi di "interpretazione" e di "esecuzione" per una singola istruzione è riportato nella figura 2.5.

L'esecuzione delle istruzioni di un programma nella esatta successione viene realizzata utilizzando le informazioni contenute nel "Registro stato del programma".

La gestione e la consultazione delle informazioni contenute in tale registro avviene nel seguente modo :

I - l'indirizzo di "lancio" viene scritto durante l'esecuzione del programma di lancio nella zona del R.S.P. (Registro Stato del Programma) che contiene l'indirizzo della successiva istruzione da eseguire.

II - alla fine del caricamento di tale indirizzo viene avviata automaticamente l'esecuzione della sequenza contenuta nella R.O.M. che realizza la "fase di interpretazione" della istruzione il cui indirizzo è contenuto nel R.S.P.

III - se l'istruzione viene "interpretata" in modo corretto la

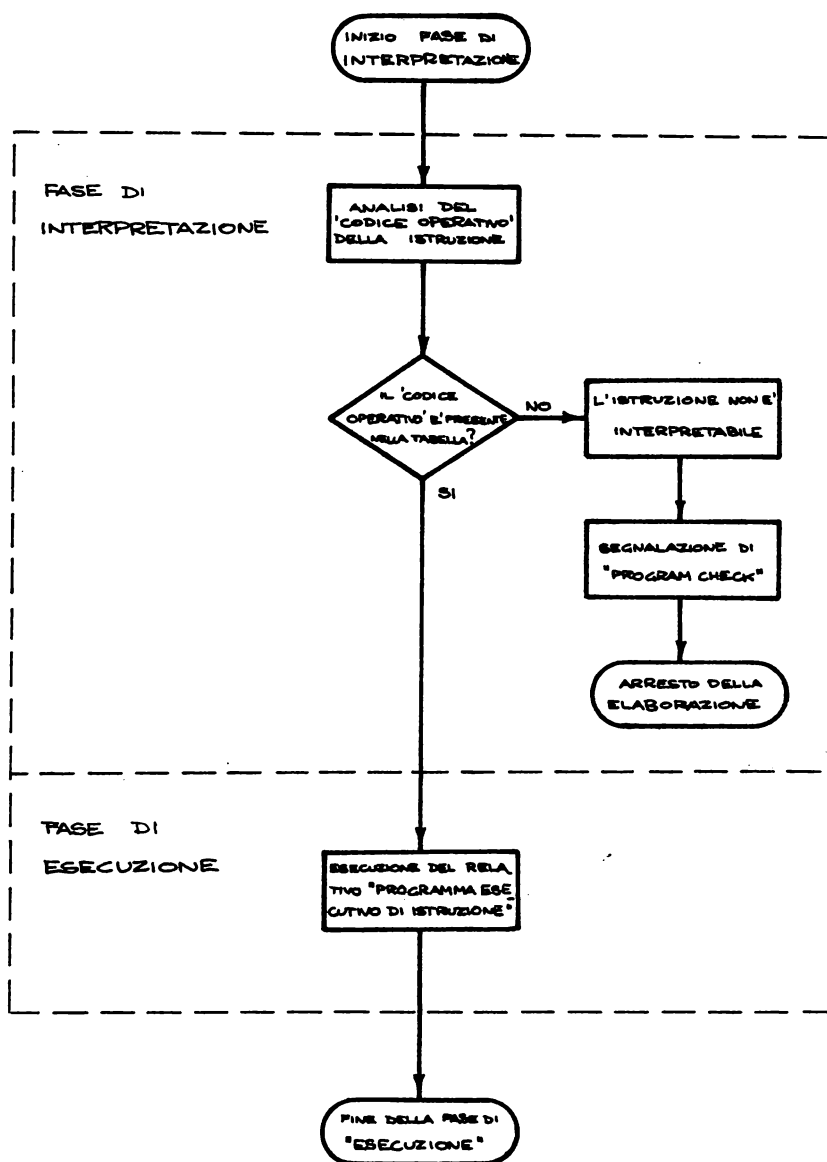



Fig. 2.5 - Schema delle fasi di "interpretazione" e di "esecuzione" di una istruzione

sua lunghezza viene scritta nella apposita zona di R.S.P. e viene sommata all'indirizzo in esso contenuto, che risulterà quindi essere quello dell'istruzione da eseguirsi successivamente.

IV - alla fine della "fase di interpretazione" viene avviata automaticamente la "fase di esecuzione" della istruzione esaminata

V - alla fine della "fase di esecuzione" il calcolatore è in grado di iniziare l'interpretazione dell'istruzione successiva, in quanto il suo indirizzo è effettivamente quello contenuto nel R.S.P., ripetendo quindi il ciclo di operazioni descritte in precedenza a partire dal punto III.

Se l'istruzione in corso di esecuzione è una istruzione di "salto" o di "salto con memorizzazione", paragrafo 3.3, prima del termine della "fase di esecuzione" l'indirizzo contenuto nel R.S.P. viene sostituito con l'indirizzo contenuto nell'istruzione stessa.

Un flow-chart di massima che descrive la successione delle operazioni svolte durante l'esecuzione di un programma, con particolare riferimento al trattamento del R.S.P., è rappresentato nella figura 2.6.

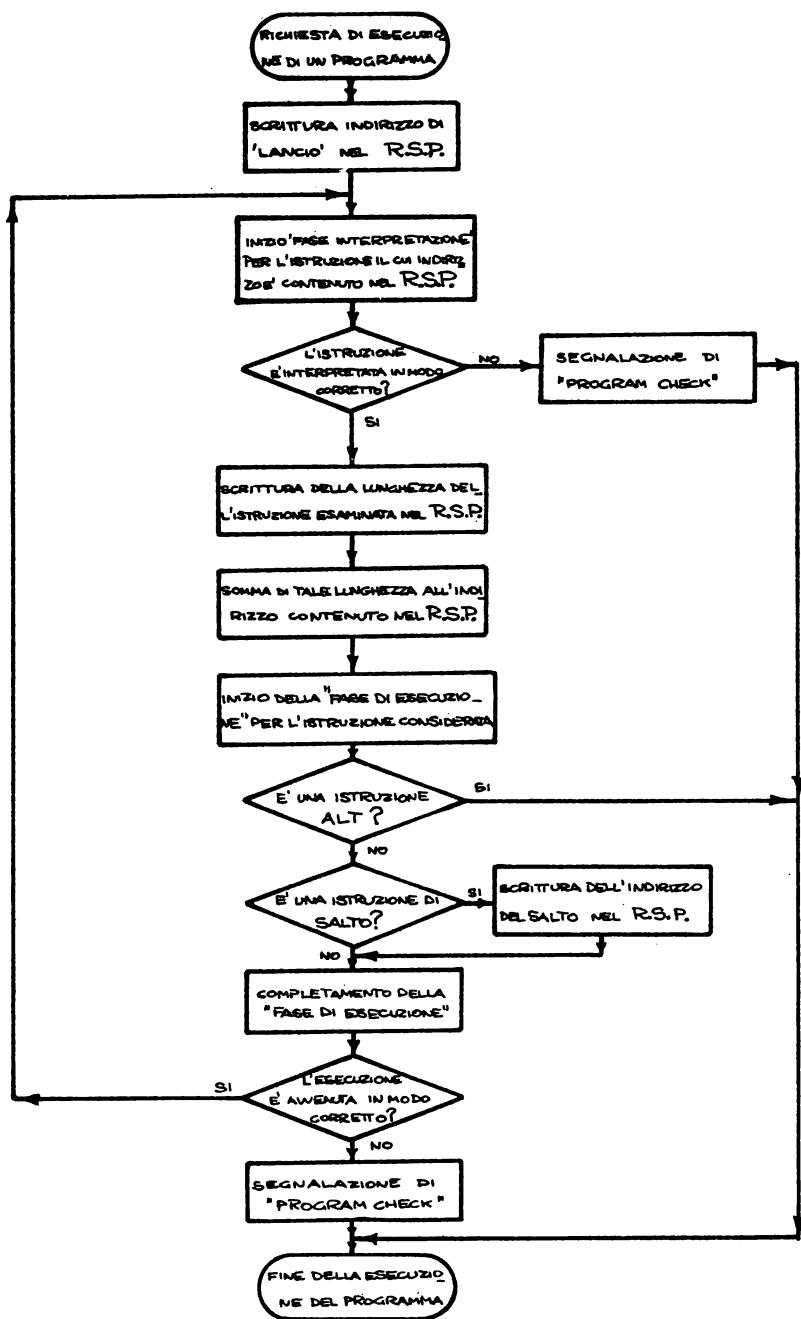



Fig. 2.6 - Schema logico dell'esecuzione di un programma

2.4 REDAZIONE DI UN PROGRAMMA

Come si è già visto nel paragrafo 2.3 tutte le istruzioni costituenti un programma vengono registrate in memoria sequenzialmente per indirizzi crescenti a partire dall'indirizzo di caricamento precedentemente specificato.

Ovviamente esse devono essere state scritte in precedenza su opportuni moduli, denominati "fogli di programmazione".

La fase di scrittura manuale di un programma sui "fogli di programmazione", che viene anche chiamata "codifica" di un programma, deve essere effettuata dopo avere completato la fase di studio e di analisi del problema, con la conseguente redazione del flow-chart o diagramma di flusso relativo al problema esaminato (paragrafo 4.1).

Nella figura 2.7 è rappresentato un "foglio di programmazione" adatto per le istruzioni del calcolatore CND.

Come si può notare in ogni riga del "foglio di programmazione" esistono 19 caselle numerate, precedute e seguite da spazi non numerati riservati alle indicazioni per i "salti" e ad eventuali "osservazioni".

In ognuna delle 19 caselle può essere scritto un solo simbolo grafico, e cioè un solo carattere dell'alfabeto esadecimale (0 ÷ 9, A ÷ F), fatta eccezione per le caselle 5,6 e 7 nelle quali possono comparire anche altri caratteri dell'alfabeto del calcolatore.

In ogni riga può essere scritta una sola istruzione.

Nelle colonne 1,2,3,4 deve essere scritto "l'indirizzo di memoria dell'istruzione" e cioè l'indirizzo del "codice operativo" dell'istruzione stessa. Come già si è detto nella versione base del calcolatore CND la colonna 1 conterrà sempre il carattere 0 poichè la memoria ha al massimo 4096 posizioni, e cioè FFF in esadecimale.

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE													OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

Fig. 2.7 - Foglio di programmazione

Le colonne 5,6,7 sono riservate al codice simbolico dell'istru - zione, il quale ha una funzione puramente mnemonica.

Nelle colonne dalla 8 alla 19 compresa devono essere scritti i caratteri componenti le istruzioni.

Per rendere minime le probabilità di errore nella scrittura ma nuale delle istruzioni queste colonne sono state suddivise con tratti più o meno marcati a seconda del significato logico dei caratteri che in essi compaiono.

Tale suddivisione è valida per il tracciato del formato A; ad essa sono comunque facilmente adattabili anche le istruzioni di formato B.

Il programma vero e proprio che deve essere caricato nel calcola tore, e cioè battuto sulla apposita tastiera di console, sarà quindi soltanto il contenuto delle colonne dalla 8 alla 19 com - prese.

Come esempio nella figura 2.8 sono riportate alcune istruzioni scritte correttamente.

Tutte le istruzioni riportate nella fig. 2.9 hanno invece un er- rore di scrittura.

Gli errori commessi sono i seguenti :

Istruzione CLO - nella colonna 10 compaiono due simboli grafici anzichè uno solo.

Istruzione SES - nelle colonne 8 ÷ 19 compare un simbolo grafico (X) non appartenente all'alfabeto esadecimale (in colonna 10).

Istruzione IAL - nella colonna 11 compaiono due simboli grafici.

Istruzione OES - l'indirizzo del 1° operando è stato scritto nel le caselle riservate all'indirizzo del 2° ope - rando.

Istruzione IAL - l'indirizzo del 1° operando non è allineato (do vrebbe iniziare dalla colonna 12)

4	CODICE SIMBOL. 5	6	7	ISTRUZIONE COD. OPER. 8	9	LUNG. 10	11	1° OPERANDO 12	13	14	15	2° OPERANDO 16	17	18	19
1	A	D	E	A	1	8	2	0	0	5	F	0	1	1	F
A	T	R	A	A	5	0	5	0	E	A	1	0	6	9	A
8	S	A	L	B	5	8	0	0	A	0	1				
9	I	E	S	B	1	2	B	0	3	8	B				
B	S	E	S	A	4	5	1	0	6	3	3	0	9	8	9
F	S	D	E	A	2	5	2	0	0	F	A	0	4	8	9
0	A	E	S	A	3	5	3	0	4	A	1	0	8	6	A

Fig. 2.8 - Esempio di istruzioni scritte in modo corretto

4	CODICE SIMBOL. 5	6	7	ISTRUZIONE COD. OPER. 8	9	LUNG. 10	11	1° OPERANDO 12	13	14	15	2° OPERANDO 16	17	18	19
2	C	L	O	A	6	05	0	0	1	8	A	0	C	1	0
5	S	E	S	A	4	X	1	0	6	3	3	0	9	1	0
A	I	A	L	B	2	3	10	0	8	A	2				
E	O	E	S	B	3	0	7					0	E	4	F
0	I	A	L	B	2	4	0		0	6	9	9			

Fig. 2.9- Esempio di istruzioni scritte in modo errato

CAPITOLO 3

DESCRIZIONE DELLE ISTRUZIONI

3.1 ISTRUZIONI ARITMETICHE E DI CONFRONTO

3.2 ISTRUZIONI DI INPUT/OUTPUT

3.3 ISTRUZIONI DI CONTROLLO

3.1 ISTRUZIONI ARITMETICHE E DI CONFRONTO

Tutte le istruzioni appartenenti a questa classe sono di formato A e quindi operano su due dati, detti "1° operando e 2° operando".

Esse danno al calcolatore la possibilità di effettuare le quattro operazioni aritmetiche fondamentali, permettono di confrontare tra di loro i dati e di effettuare trasferimenti nell'ambito della memoria.

Occorre tenere presente che essendo possibili due classi di dati, alfanumerici ed esadecimali, è necessario controllare sempre il tipo di rappresentazione interna dei dati sui quali si opera in quanto tutte le istruzioni aritmetiche e quella di "confronto algebrico" operano soltanto su dati entrambi dello stesso tipo.

Nel caso che i dati non siano del tipo richiesto dalla istruzione o non siano entrambi dello stesso tipo, l'operazione non viene eseguita e l'esecuzione del programma viene arrestata.

Descriviamo ora dettagliatamente le varie funzioni ed i formati delle istruzioni appartenenti a questa classe.

3.1.1 ADDIZIONE DECIMALE

Funzione

Questa istruzione è di formato A ed opera sul contenuto di 2 campi di memoria detti rispettivamente "primo e secondo operando".

Essa provoca la somma decimale algebrica del dato secondo operando al dato primo operando; il risultato viene posto nel campo primo operando.

Gli operandi devono essere di tipo decimale, entrambi segnati o entrambi non segnati; in caso contrario l'istruzione non viene eseguita.

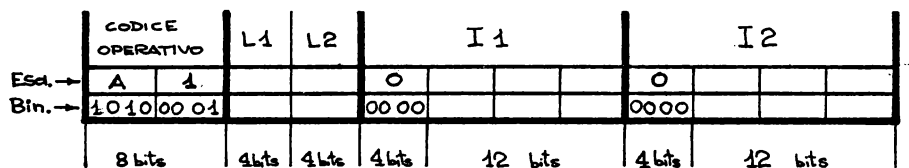
La lunghezza del secondo operando può essere minore od uguale a quella del primo operando.

Se è minore il secondo operando viene considerato, agli effetti della somma, della stessa lunghezza del primo, con l'ideale aggiunta di zeri non significativi.

Il risultato della somma può non essere contenuto nel campo primo operando, dando così luogo al caso di "overflow". In tal caso le cifre eccedenti vengono perdute e quindi il risultato della somma è errato.

Il verificarsi di tale evenienza è indicata dal valore binario 11 del Codice di Condizione (paragrafo 1.9).

Se la lunghezza del secondo operando è superiore a quella del primo, l'istruzione non viene eseguita.

	CODICE OPERATIVO		L1	L2	I1				I2			
Esa. →	A	1			0				0			
Bin. →	1010	0001			0000				0000			
	8 bits		4 bits	4 bits	4 bits	12 bits			4 bits	12 bits		

Codice operativo = A1 CODICE SIMBOLICO = ADE

L1 = lunghezza, in bytes, del primo operando diminuita di 1 (≤ 15, carattere esadecimale F)

L2 = lunghezza, in bytes, del secondo operando diminuita di 1 (≤ 15, carattere esadecimale F)

I1 = indirizzo del primo operando

I2 = indirizzo del secondo operando

Fig. 3.1 - Formato della "addizione decimale"

Codice Condizione

I valori binari assunti dal Codice di Condizione hanno i seguenti significati

00 se il risultato della somma è = 0

01 se il risultato della somma è < 0

10 se il risultato della somma è > 0

11 se si è verificato il caso di OVERFLOW.

Esempio

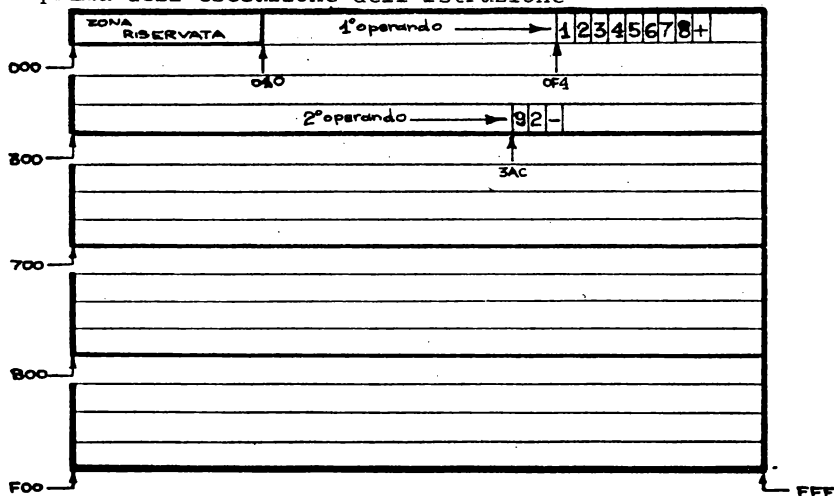
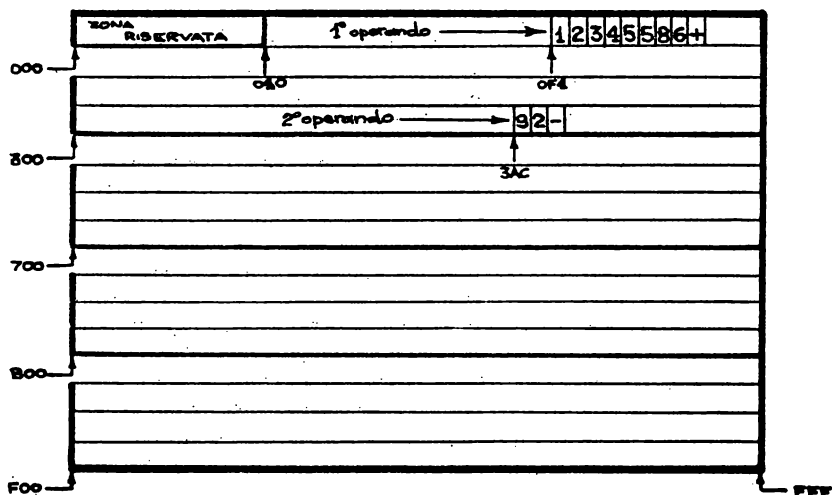
La seguente istruzione :

A	1	8	2	0	0	F	4	0	3	A	C

fa eseguire la somma di un dato lungo 3 caratteri che si trova memorizzato all'indirizzo 3AC (940 in decimale) con un dato lungo 9 caratteri che si trova memorizzato all'indirizzo 0F4 (240 in decimale).

Il risultato di tale somma viene posto all'indirizzo 0F4.

Le mappe della memoria con il contenuto delle zone interessate (1° operando e 2° operando, entrambi di tipo decimale segnato), prima e dopo l'esecuzione delle istruzioni sono le seguenti :

a - prima dell'esecuzione dell'istruzione

b - dopo l'esecuzione dell'istruzione

Il codice di condizione assume il valore binario 10 in quanto il risultato della somma è maggiore di zero.

3.1.2 SOTTRAZIONE DECIMALE

Funzione

Il formato di questa istruzione è del tipo A; essa opera sul contenuto di due campi di memoria, detti rispettivamente "primo e secondo operando".

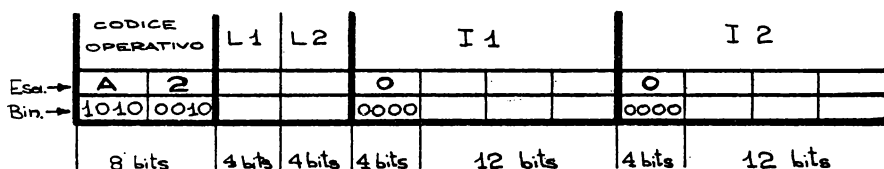
Gli operandi devono essere di tipo decimale, entrambi segnato o entrambi non segnati.

Il dato secondo operando è sottratto in modo algebrico dal primo; il risultato della sottrazione viene posto nel campo primo operando.

Anche in questa istruzione, come nella ADDIZIONE DECIMALE, la lunghezza del secondo operando non deve superare quella del primo; in caso contrario l'istruzione non viene eseguita.

Il caso di "overflow" si verifica quando il risultato della sottrazione non può essere contenuto nel campo primo operando, oppure quando i dati sono entrambi non segnati ed il secondo operando ha valore maggiore del primo. In quest'ultimo caso il risultato che viene a trovarsi nel campo primo operando è il complemento del valore assoluto della differenza rispetto alla potenza di 10 immediatamente superiore.

In caso di overflow il Codice di Condizione assume il valore binario 11.

	CODICE OPERATIVO		L 1	L 2	I 1				I 2			
Esa.→	A	2			0				0			
Bin.→	1010	0010			0000				0000			
	8 bits		4 bits	4 bits	4 bits	12 bits			4 bits	12 bits		

Codice operativo = A 2 CODICE SIMBOLICO = SDE

L1 = lunghezza in bytes del primo operando diminuita di 1 (≤ 15, carattere esadecimale F)

L2 = lunghezza in bytes del secondo operando diminuita di 1 (≤ 15, carattere esadecimale F)

I1 = indirizzo del primo operando

I2 = indirizzo del secondo operando

Fig. 3.2 - Formato della "sottrazione decimale"

Codice Condizione

I valori binari assunti dal Codice di Condizione hanno i seguenti significati

00 se il risultato della sottrazione è = 0

01 se il risultato della sottrazione è < 0 (operandi segnati)

10 se il risultato della sottrazione è > 0

11 se si è verificato il caso di OVERFLOW

Esempio

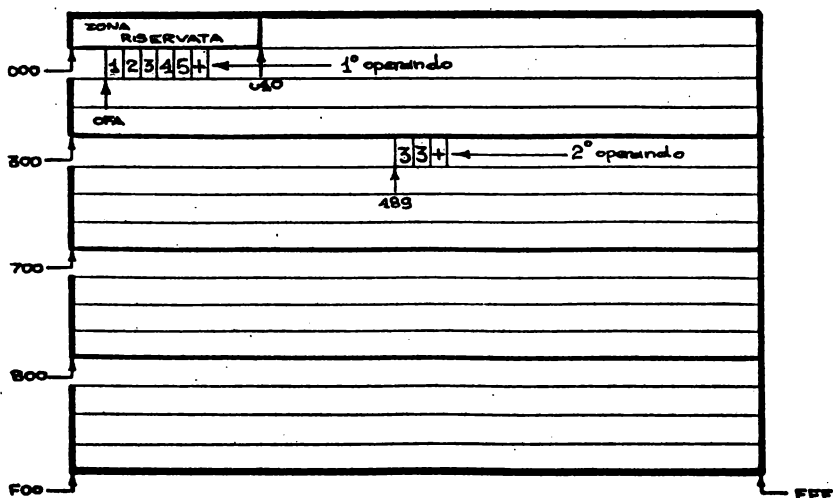
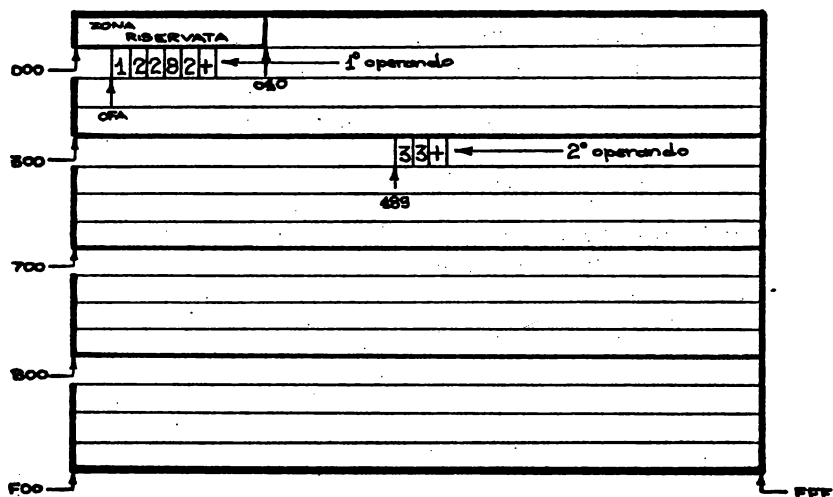
La seguente istruzione

A	2	5	2	0	0	F	A	0	4	8	9

provoca l'esecuzione della sottrazione di un dato lungo 3 caratteri memorizzato all'indirizzo 489 (1161 in decimale) da un dato lungo 6 caratteri memorizzato all'indirizzo OFA (266 in decimale).

Il risultato della sottrazione viene posto nel campo di indirizzo OFA.

Le mappe della memoria con il contenuto delle zone interessate (1° operando e 2° operando entrambi di tipo decimale segnato) prima e dopo l'esecuzione della istruzione sono le seguenti :

a - prima dell'esecuzione

b - dopo l'esecuzione della istruzione

Il Codice di Condizione assume il valore binario 10 in quanto il risultato della sottrazione è maggiore di zero.

3.1.3 ADDIZIONE ESADECIMALE

Funzione

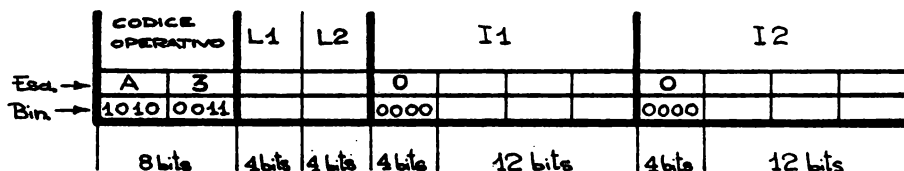
Il formato di questa istruzione è del tipo A; essa opera sul contenuto di due campi di memoria detti rispettivamente "primo e secondo operando".

I due operandi devono essere numeri esadecimali.

Il dato secondo operando è sommato in modo esadecimale al primo; il risultato della somma viene posto nel campo primo operando.

Per la lunghezza dei due operandi valgono le stesse regole già esposte nella descrizione dell'ADDIZIONE DECIMALE.

I valori che compaiono nei semibytes L1 e L2 devono essere sempre dispari, ed esprimono il numero delle cifre esadecimali dei due operandi diminuito di 1.

	CODICE OPERATIVO		L1	L2	I1				I2			
Esa. →	A	3			0				0			
Bin. →	1010	0011			0000				0000			
	8 bits		4 bits	4 bits	4 bits	12 bits			4 bits	12 bits		

Codice Operativo = A3 CODICE SIMBOLICO = AES

L1	= lunghezza, in semibytes, del primo operando diminuita di 1 (≤ 15, carattere esadecimale F)
L2	= lunghezza, in semibytes, del secondo operando diminuita di 1 (≤ 15 carattere esadecimale F)
I1	= indirizzo del primo operando
I2	= indirizzo del secondo operando

Fig. 3.3 - Formato della "addizione esadecimale"

Codice Condizione

I valori binari assunti dal Codice di Condizione hanno i seguenti significati :

00 se il risultato della somma è = 0

10 se il risultato è > 0

11 se si è verificato il caso di OVERFLOW.

Il valore 01 non può essere assunto dal codice di condizione dopo una addizione esadecimale.

Esempio

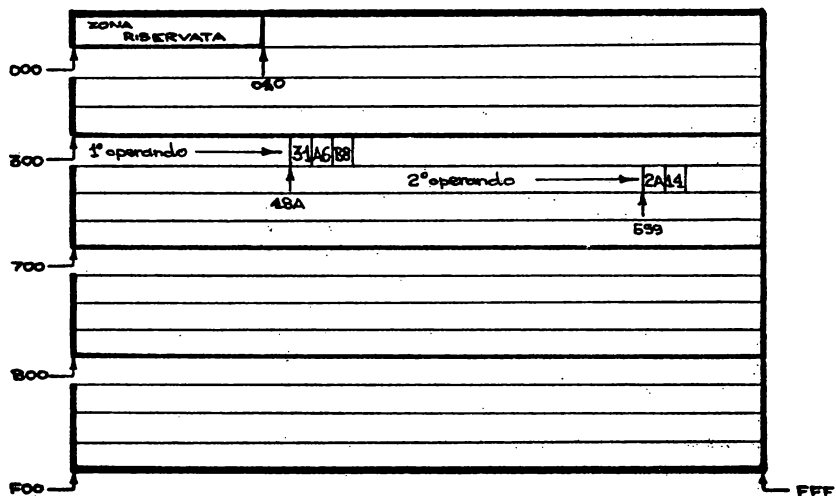
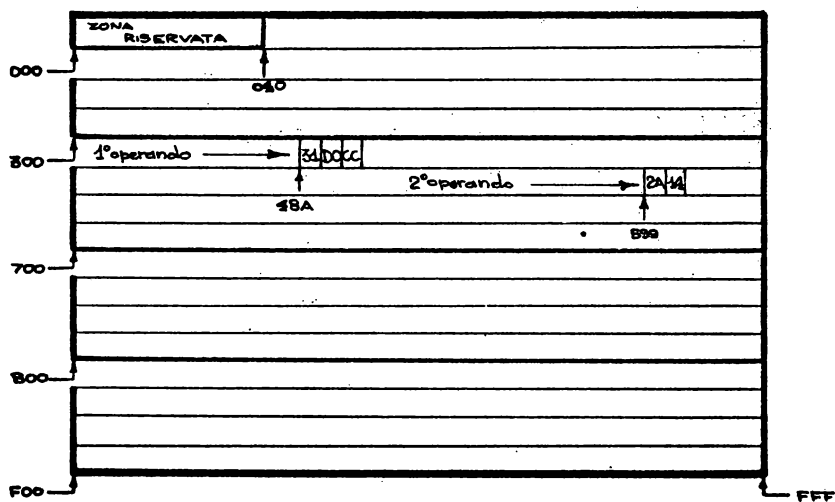
La seguente istruzione :

A	3	5	3	0	4	8	A	0	5	9	9

provoca la somma del dato lungo 4 cifre esadecimali che si trova memorizzato all'indirizzo 599 (1433 in decimale) con il dato lungo 6 cifre esadecimali che si trova memorizzato all'indirizzo 48A (1162 in decimale).

Il risultato di tale somma viene posto nel campo di indirizzo 48A.

Le mappe della memoria con il contenuto delle zone interessate (1° operando e 2° operando) prima e dopo l'esecuzione della istruzione sono le seguenti :

a - prima dell'esecuzione della istruzione

b - dopo l'esecuzione della istruzione

Il "codice di condizione" assume il valore binario 10 in quanto la somma è avvenuta in modo corretto.

3.1.4 SOTTRAZIONE ESADECIMALE

Funzione

Il formato di questa istruzione è del tipo A; essa opera sul contenuto di due campi di memoria detti rispettivamente "primo e secondo operando".

I due operandi devono essere numeri esadecimali.

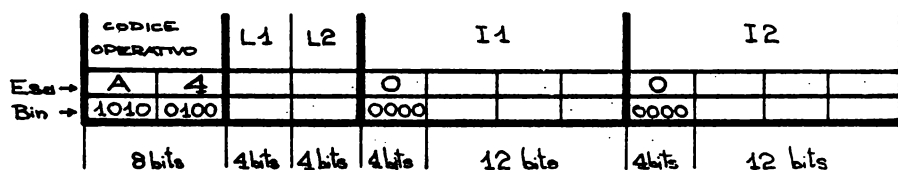
Il dato secondo operando è sottratto in modo esadecimale dal dato primo operando; il risultato della sottrazione viene posto nel campo primo operando.

Per la lunghezza dei due operandi valgono le stesse regole già esposte nella descrizione dell'ADDIZIONE DECIMALE.

La sottrazione viene eseguita sommando al 1° operando il complemento binario aumentato di 1 del 2° operando (immaginato della stessa lungheza del 1° con l'aggiunta ideale di zeri non significativi) e non considerando i riporti al di fuori delle lunghezze dei due operandi.

Se il valore del 1° operando è maggiore od uguale a quello del 2° si ottiene la differenza fra i due, mentre se il valore del 1° operando è minore di quello del 2° si ottiene il complemento binario della differenza, in quanto il valore effettivo della differenza sarebbe negativo. In quest'ultimo caso si ha la segnalazione di Overflow.

I valori che compaiono nei semibytes L1 e L2 devono essere sempre dispari, ed esprimono il numero delle cifre esadecimali dei due operandi diminuito di 1.

	CODICE OPERATIVO		L1	L2	I1		I2	
Esa →	A	4			0		0	
Bin →	1010	0100			0000		0000	
	8 bits		4 bits	4 bits	4 bits	12 bits	4 bits	12 bits

Codice operativo = A4 CODICE SIMBOLICO = SES

L1 = lunghezza in semibytes del primo operando diminuita di 1 (≤ 15, carattere esadecimale F)

L2 = lunghezza in semibytes del secondo operando diminuita di 1 (≤ 15, carattere esadecimale F)

I1 = indirizzo del primo operando

I2 = indirizzo del secondo operando

Fig.3.4 - Formato della "sottrazione esadecimale"

Codice Condizione

I valori binari assunti dal Codice di Condizione hanno i seguenti significati :

00 il risultato della sottrazione è = 0

10 il risultato della sottrazione è > 0

11 se si è verificato il caso di overflow.

Il valore 01 non può essere assunto dal Codice di Condizione dopo una sottrazione esadecimale.

Esempio

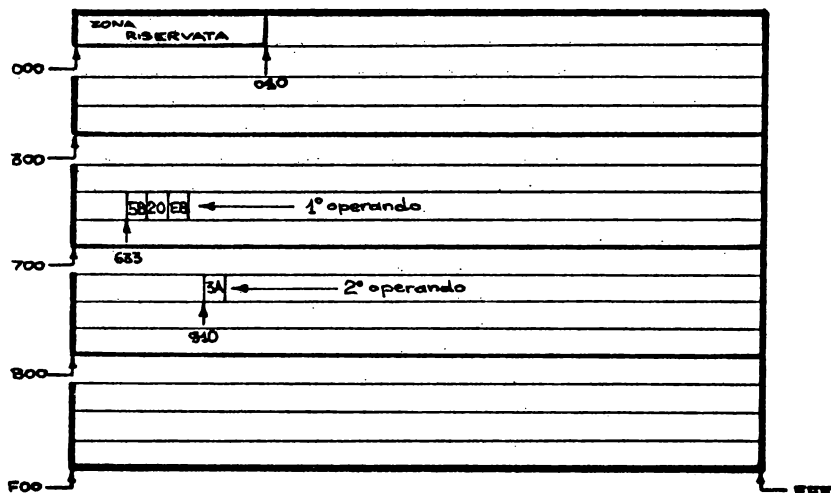
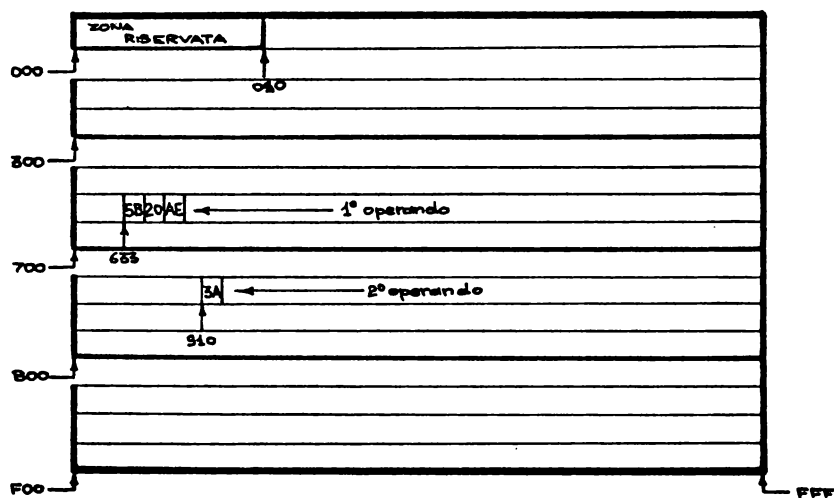
La seguente istruzione

A	4	5	1	0	6	3	3	0	9	1	0

provoca la sottrazione del dato di tipo esadecimale lungo 2 ci-
fre che si trova memorizzato all'indirizzo 910 (2320 in decimale)
dal dato di tipo esadecimale lungo 6 cifre che si trova memoriz-
zato all'indirizzo 633 (1587 in decimale).

Il risultato di tale differenza viene posto nel campo di indiriz-
zo 633.

Le mappe della memoria con il contenuto delle zone interessate
(1° operando e 2° operando) prima e dopo l'esecuzione della istru-
zione sono le seguenti

a - prima dell'esecuzione della istruzione

b - dopo l'esecuzione della istruzione

Il "Codice di Condizione" assume il valore binario 10 in quanto il risultato della sottrazione è maggiore di zero.

3.1.5 TRASFERIMENTO

Funzione

Il formato di questa istruzione è del tipo A; essa opera sul contenuto di 2 campi di memoria detti rispettivamente "primo e secondo operando".

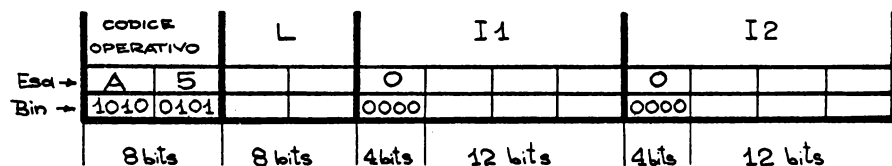
Essa provoca il trasferimento del contenuto di una zona di memoria in un'altra zona di memoria.

La lunghezza della zona da trasferire, e cioè del secondo operando, è quella indicata nella istruzione e coincide ovviamente con quella del primo operando.

I dati del campo "secondo operando" vengono trasferiti nel campo "primo operando" serialmente a partire dalla sinistra.

Occorre notare che in questa istruzione, pur essendo del tipo A, viene specificata una sola lunghezza; il valore espresso, per tale lunghezza, può raggiungere 255 essendo disponibili 8 bits.

Pertanto il formato di questa istruzione viene anche detto formato A ad una lunghezza. Il contenuto del 2° operando non viene alterato dall'esecuzione di questa istruzione.

	CODICE OPERATIVO		L		I1				I2			
Esa →	A	5			0				0			
Bin →	1010	0101			0000				0000			
	8 bits		8 bits		4bits	12 bits			4bits	12 bits		

Codice Operativo = A5 CODICE SIMBOLICO = TRA

L = lunghezza in bytes della zona da trasferire diminuita di 1 ($\leq$ 255, FF in esadecimale)

I1 = indirizzo della zona di arrivo del "secondo operando"

I2 = indirizzo della zona da trasferire

Fig. 3.5 - Formato del "Trasferimento"

Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

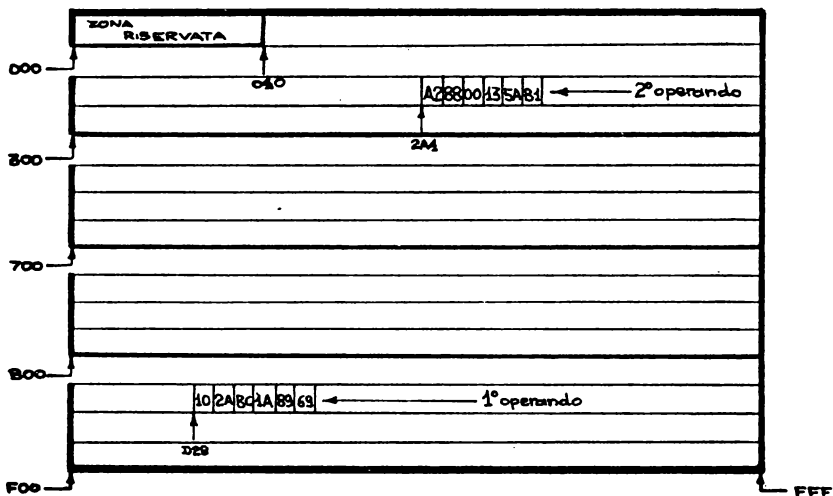
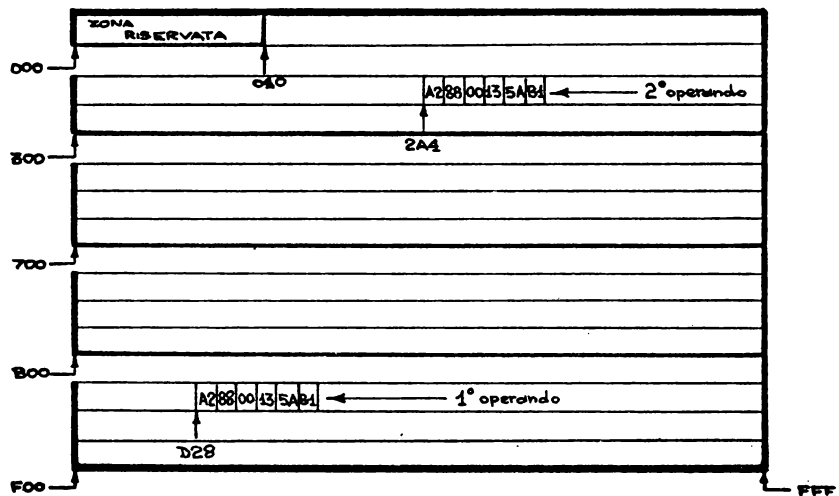
Esempio

La seguente istruzione

A5	05	0D	28	02	A4

provoca il trasferimento del contenuto della zona di memoria lunga 6 bytes di indirizzo 2A4 (676 in decimale) nella zona di memoria di indirizzo D28 (3368 in decimale).

Le mappe della memoria con il contenuto delle zone interessate (1° operando e 2° operando) prima e dopo l'esecuzione della istruzione sono le seguenti

a - prima dell'esecuzione della istruzione

b - dopo l'esecuzione della istruzione

3.1.6 CONFRONTO LOGICO

Funzione

Il formato di questa istruzione è del tipo A ad una lunghezza; essa opera sul contenuto di due campi di memoria detti rispettivamente "primo e secondo operando".

Essa provoca la comparazione binaria fra i due campi di memoria.

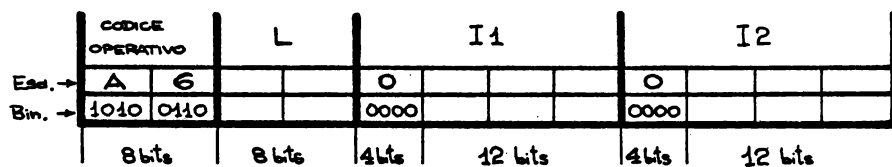
I dati contenuti nei campi vengono considerati in formato esadecimale; cioè viene effettuato il confronto del contenuto esadecimale di un byte di un campo con il contenuto esadecimale del byte corrispondente nell'altro campo.

Le cifre esadecimali vengono considerate successivamente a partire dalla sinistra di ogni campo verso destra.

I due campi sono considerati di uguale lunghezza che è quella specificata in L (max 256 caratteri).

L'operazione di confronto viene arrestata non appena due bytes confrontati risultano diversi o quando tutti i bytes dei due operandi sono stati considerati.

L'esecuzione di questa istruzione non modifica gli operandi ma altera il valore del codice di condizione.

	CODICE OPERATIVO		L		I1				I2			
Esa. →	A	6			0				0			
Bin. →	1010	0110			0000				0000			
	8 bits		8 bits		4 bits	12 bits			4 bits	12 bits		

Codice operativo = A6 CODICE SIMBOLICO = CLO

L = lunghezza, in bytes, diminuita di 1 dei dati da confrontare (≤ 255, FF in esadecimale)

I1 = indirizzo del primo operando del confronto

I2 = indirizzo del secondo operando del confronto.

Fig. 3.6 - Formato del "Confronto Logico"

Codice Condizione

I valori binari assunti dal "codice di condizione" hanno i seguenti significati :

00 se il primo operando è uguale al secondo operando

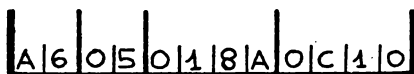
01 se il primo operando è minore del secondo operando

10 se il primo operando è maggiore del secondo operando

Il valore 11 non può essere assunto dal codice di condizione dopo un confronto logico.

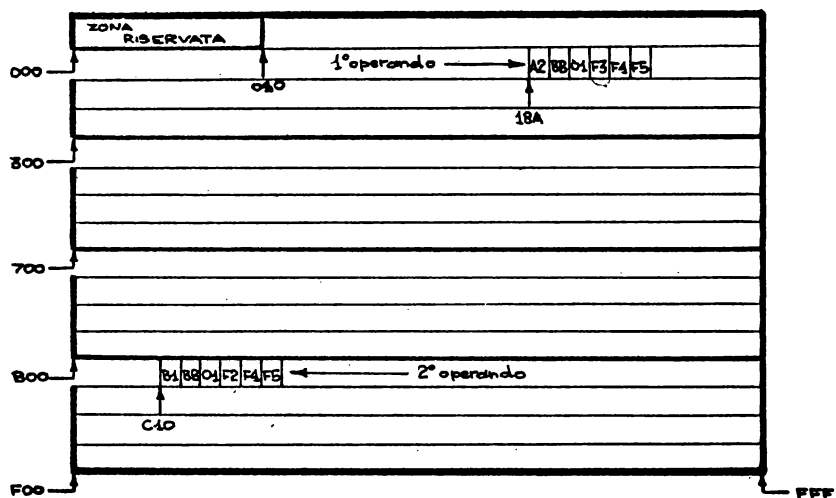
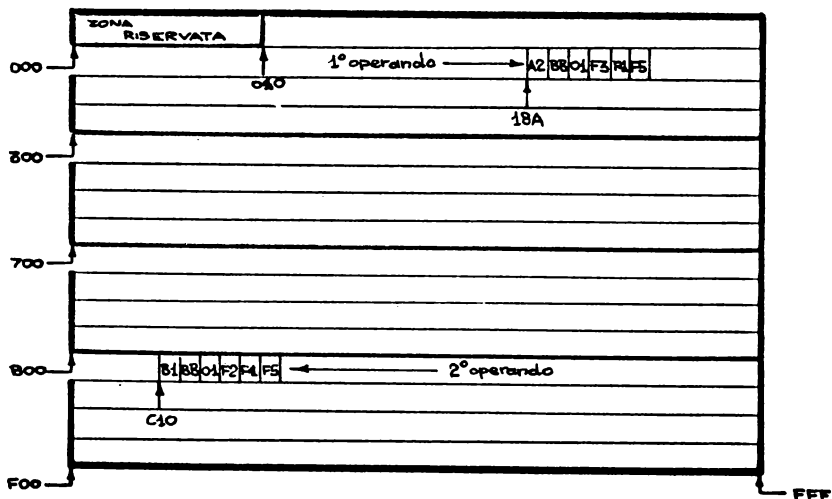
Esempio

La seguente istruzione :

A	6	0	5	0	1	8	A	0	C	1	0

provoca il confronto logico del contenuto dei 6 bytes di indirizzo 018A (in decimale 394) con il contenuto dei 6 bytes che si trovano all'indirizzo 0C10 (in decimale 3088).

Il risultato del confronto imposta opportunamente il codice di condizione.

Le mappe della memoria con il contenuto delle zone interessate (1° operando e 2° operando) prima e dopo l'esecuzione della istruzione sono le seguenti :

a - prima dell'esecuzione della istruzione

b - dopo l'esecuzione della istruzione

Il risultato di tale "confronto logico" è minore. Infatti la pri
ma cifra esadecimale del 1° operando è un A, mentre la corrispon
dente cifra del 2° operando è B; il "codice di condizione" assu-
me il valore "01". Come si può notare i due schemi precedenti so
no perfettamente identici: questo poichè l'istruzione non modifi
ca il contenuto delle zone di memoria, ma si limita ad effettuar
ne il confronto.

3.1.7 CONFRONTO ALGEBRICO

Funzione

Il formato di questa istruzione è del tipo A; essa opera sul contenuto di due campi di memoria detti rispettivamente "primo e secondo operando" effettuandone il confronto algebrico.

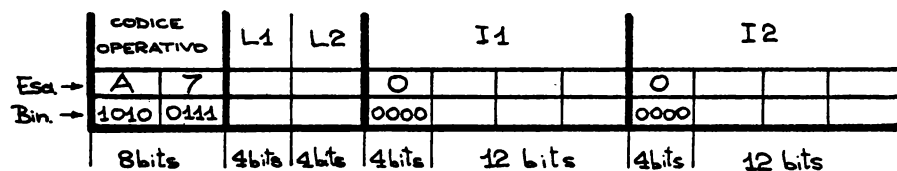
I dati contenuti nei due campi devono essere del tipo decimale, entrambi segnati od entrambi privi di segno; in caso contrario l'istruzione non viene eseguita.

Il più corto dei due dati è considerato, agli effetti del confronto, espanso con zeri non significativi sino a raggiungere la lunghezza dell'altro.

I due dati possono essere lunghi al massimo 16 caratteri; le loro lunghezze sono espresse nei semibytes L1 e L2.

I caratteri decimali vengono confrontati successivamente a partire dalla sinistra di ciascun campo; se i dati sono segnati viene preventivamente effettuato il confronto dei segni. L'operazione di confronto viene arrestata non appena due caratteri confrontati risultano diversi oppure quando tutti i caratteri dei due operandi sono stati considerati.

L'esecuzione di questa istruzione non modifica i due operandi, ma altera il valore del codice di condizione.

	CODICE OPERATIVO		L1	L2	I1				I2			
Esa →	A	7			0				0			
Bin. →	1010	0111			0000				0000			
	8bits		4bits	4bits	4bits	12 bits			4bits	12 bits		

Codice operativo = A7 CODICE SIMBOLICO = CAL

L1 = lunghezza in bytes, diminuita di 1 del primo dato da confrontare ($\leq$ 15, F in esadecimale)

L2 = lunghezza in bytes, diminuita di 1 del secondo dato da confrontare ($\leq$ 15, F in esadecimale)

I1 = indirizzo del primo operando del confronto

I2 = indirizzo del secondo operando del confronto.

Fig. 3.7 - Formato del "Confronto Algebrico"

Codice Condizione

I valori binari assunti dal "codice di condizione" hanno i seguenti significati :

00 se il primo operando è uguale al secondo operando

01 se il primo operando è minore del secondo operando

10 se il primo operando è maggiore del secondo operando.

Il valore 11 non può essere assunto dal codice di condizione dopo un confronto logico.

Esempio

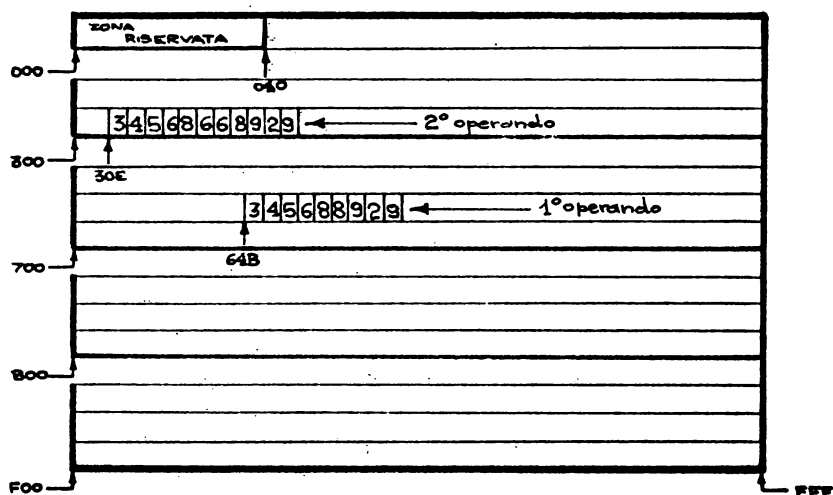
La seguente istruzione :

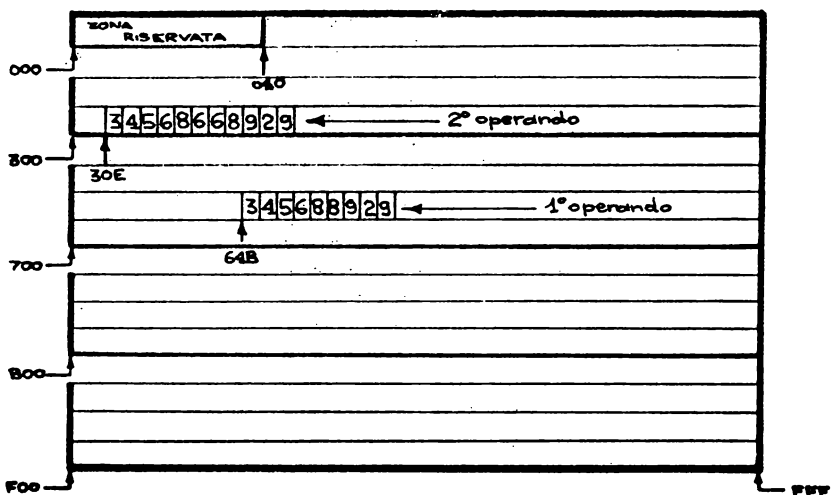
A	7	8	A	0	6	4	B	0	3	0	E

provoca il confronto algebrico del dato lungo 9 bytes, memorizzato all'indirizzo 64B (1611 in decimale) con il dato lungo 11 bytes, memorizzato all'indirizzo 30E (782 in decimale).

Il risultato del confronto imposta opportunamente il codice di condizione.

Le mappe della memoria con il contenuto delle zone interessate (1° operando e 2° operando) prima e dopo l'esecuzione della istruzione sono le seguenti :

a - prima dell'esecuzione della istruzione

I dati sono entrambi di tipo decimale non segnato, pertanto l'istruzione viene eseguita.

b - dopo l'esecuzione della istruzione

Il valore del 1° operando è "minore" del 2° operando, e quindi il "codice di condizione" assume il valore "1".

Come per l'istruzione, si può notare che il contenuto delle zone di memoria interessate non viene modificato dall'istruzione.

3.2 ISTRUZIONI DI INPUT-OUTPUT

Tutte le istruzioni appartenenti a questo gruppo sono di formato B e contengono pertanto l'indirizzo di un solo operando.

Esse permettono al calcolatore di ricevere informazioni tramite una delle due tastiere di cui è dotato o di stampare informazioni sull'unità di stampa.

Come si è detto in precedenza il calcolatore CND è dotato di una tastiera esadecimale che fa parte del quadro di comando e sulla quale è possibile "digitare" soltanto cifre esadecimali e di una tastiera alfanumerica incorporata nella unità di input-output sulla quale è possibile "digitare" uno qualsiasi dei 64 caratteri dell'alfabeto del calcolatore.

L'uso di ciascuna delle due tastiere viene abilitato da due distinte istruzioni di input che verranno descritte dettagliatamente nel seguito insieme con le due istruzioni di output che permettono la stampa di informazioni esadecimali od alfanumeriche.

3.2.1 INTRODUZIONE ESADECIMALE

Funzione

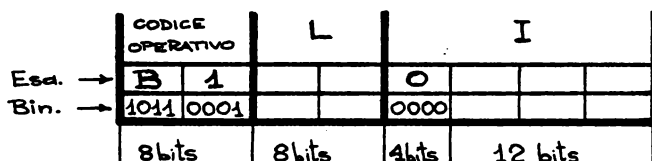
Il formato di questa istruzione è del tipo B.

Essa abilita il calcolatore a ricevere dati di tipo esadecimale dalla tastiera di console provocando lo sblocco di questa e la accensione della LAMPADA 'ESA' in console.

Il numero delle cifre esadecimali che devono essere introdotte, diminuito di una unità, viene specificato in L .

La LAMPADA ESA rimane accesa e la tastiera di console rimane sbloccata, sino a che il numero delle cifre esadecimali introdotte non raggiunge il valore voluto.

Le cifre esadecimali così introdotte, il cui numero massimo è 256, vengono scritte in memoria a partire dall'indirizzo specificato in I.

Come si è detto il numero delle cifre esadecimali che compare in L è quello effettivo diminuito di 1, quindi il valore di L dovrà essere sempre dispari (numero effettivo sempre pari); in caso contrario l'istruzione non viene eseguita.

Codice operativo = B1 CODICE SIMBOLICO = IES

L = lunghezza, in semibytes, del dato da introdurre diminuita di 1 (≤ 255, FF in esadecimale)

I = indirizzo a partire dal quale vengono scritti in memoria i caratteri esadecimali

Fig. 3.8 - Formato della "Introduzione esadecimale"

Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

Segnalazioni luminose in console

Quando viene incontrata questa istruzione si accende in console la LAMPADA "ESA", che si spegnerà soltanto alla fine della sua esecuzione.

Inoltre il contenuto binario dei semibytes interessati viene visualizzato sugli otto lampadini affiancati a tale lampada.

Al momento dell'introduzione di ciascun semibytes di sinistra i lampadini corrispondenti al semibytes di destra vengono spenti.

Esempio

L'istruzione IES viene generalmente utilizzata per introdurre in memoria delle costanti esadecimali, o per modificare o correggere una istruzione di un programma già esistente in memoria.

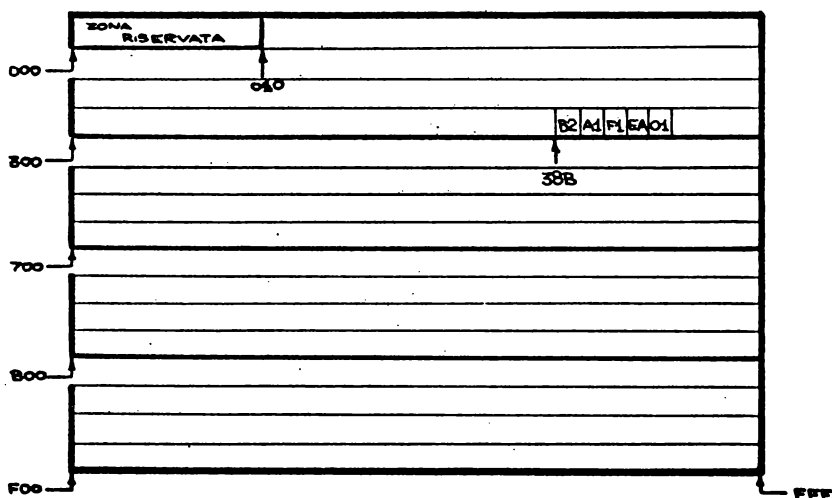
Ad esempio la seguente istruzione :

B	1	0	9	0	3	8	B

sblocca la tastiera della console per l'introduzione di 10 cifre esadecimali che verranno scritte in memoria a partire dall'indirizzo 38B (907 in decimale).

La mappa della memoria dopo l'introduzione delle 10 cifre esadecimali

B2A1F1EA01

è la seguente :

3.2.2 INTRODUZIONE ALFANUMERICA

Funzione

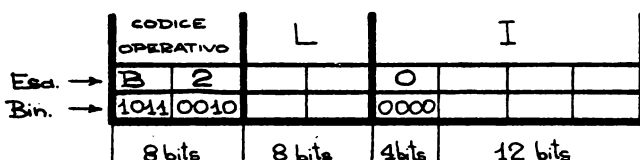
Il formato di questa istruzione è del tipo B.

Essa abilita il calcolatore a ricevere dati di tipo alfanumerico dalla tastiera della telescrivente provocando lo sblocco di questo e l'accensione della LAMPADA "ALF" di console.

Il numero dei caratteri che devono essere introdotti diminuito di una unità viene specificato in L. La LAMPADA "ALF" rimane accesa sino a che il numero dei caratteri introdotti non raggiunge il valore voluto.Con lo spegnimento della LAMPADA "ALF", si ha anche contemporaneamente il blocco meccanico della tastiera.

I caratteri alfanumerici introdotti vengono scritti in memoria a partire dall'indirizzo indicato in I, secondo la codifica riportata nella tabella del paragrafo 1.3.

Il numero massimo di caratteri alfanumerici che possono essere introdotti con questa istruzione è 256.

	CODICE OPERATIVO		L		I			
Esa. →	B	2			0			
Bin. →	1011	0010			0000			
	8 bits		8 bits		4 bits	12 bits		

Codice operativo = B2 CODICE SIMBOLICO = IAL

L = lunghezza, in bytes, del dato da introdurre diminuita di 1 (≤ 255,FF in esadecimale)

I = indirizzo a partire dal quale vengono scritti in memoria i caratteri.

Fig. 3.9 - Formato della "Introduzione Alfanumerica"

Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

Segnalazioni luminose in console

Quando viene incontrata questa istruzione si accende in console la LAMPADA "ALF", che verrà spenta soltanto alla fine della sua esecuzione. Inoltre la codificazione binaria dei caratteri in - trodotti viene visualizzata sugli otto lampadini affiancati al- la lampada stessa.

Esempio

La seguente istruzione

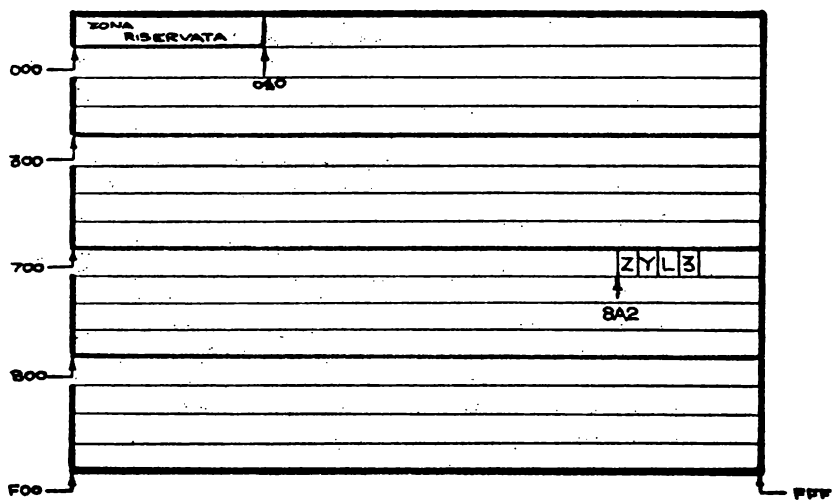
B	2	0	3	0	8	A	2

sblocca la tastiera della telescrivente per l'introduzione di 4 caratteri alfanumerici che verranno scritti in memoria a partire dall'indirizzo 8A2 (2210 in decimale).

La mappa della memoria, dopo l'introduzione dei 4 caratteri alfa numerici

ZYL3

è la seguente

ZONA RISERVATA
000
050
300
700
ZYL3
8A2
800
F00
FFF

3.2.3 ESTRAZIONE ESADECIMALE

Funzione

Il formato di questa istruzione è del tipo B.

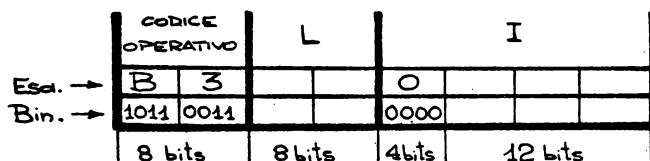
Essa provoca la stampa di un determinato numero di cifre esadecimali che sono memorizzate a partire dall'indirizzo specificato in I.

Il numero specificato in L è il numero delle cifre esadecimali che devono essere stampate, diminuite di uno, ed il suo valore dovrà essere sempre un numero dispari; in caso contrario l'istruzione non viene eseguita.

Il numero di cifre esadecimali che possono essere stampate con questa istruzione è 256. I caratteri vengono stampati serialmente da sinistra verso destra in modo consecutivo; il ritorno a capo e l'interlinea avvengono in modo automatico nei seguenti casi :

a) - prima dell'esecuzione della istruzione

b) - quando la riga di stampa è completa.

	CODICE OPERATIVO		L		I			
Esa. →	B	3			0			
Bin. →	1011	0011			0000			
	8 bits		8 bits		4 bits	12 bits		

Codice operativo = B3 CODICE SIMBOLICO=OES

L = lunghezza, in semibytes, diminuita di 1 del dato che deve essere stampato($\leqslant$ 255, cioè FF in esadecimale)

I = indirizzo del dato da stampare.

Fig. 3.10 - Formato della "Estrazione esadecimale"

Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

Esempio

La seguente istruzione :

B	3	0	7	0	E	4	F

provoca la stampa di 8 cifre esadecimali che si trovano in memoria a partire dall'indirizzo E4F (3663 in decimale).

Se il contenuto della memoria prima dell'esecuzione della istruzione è il seguente :

verrà effettuata la seguente stampa :

E182 4A FA

3.2.4. ESTRAZIONE ALFANUMERICA

Funzione

Il formato di questa istruzione è del tipo B.

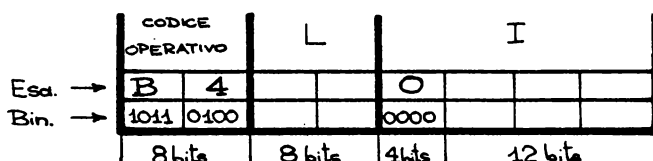
Essa provoca la stampa di un determinato numero di caratteri alfanumerici che sono memorizzati a partire dall'indirizzo specificato in I.

Il numero di caratteri alfanumerici che devono essere stampati diminuito di 1 è specificato in L e non può superare il valore 256.

La stampa dei caratteri avviene in modo seriale da sinistra verso destra; il ritorno a capo e l'interlinea avvengono in modo automatico nei seguenti casi :

a) - prima dell'esecuzione dell'istruzione

b) - quando la riga di stampa è completa.

Se nella zona di memoria contenente i caratteri da stampare sono compresi caratteri non appartenenti all'alfabeto del calcolatore essi vengono sostituiti con "blank".

Codice operativo = B4 CODICE SIMBOLICO = OAL

L = lunghezza diminuita di 1 del dato che deve essere stampato, in bytes (≤ 255, FF in esadecimale)

I = indirizzo del dato da stampare.

Fig. 3.11 - Formato della "Estrazione Alfanumerica"

Codice Condizione

Il codice di condizione non viene gestito da questa istruzione

Esempio

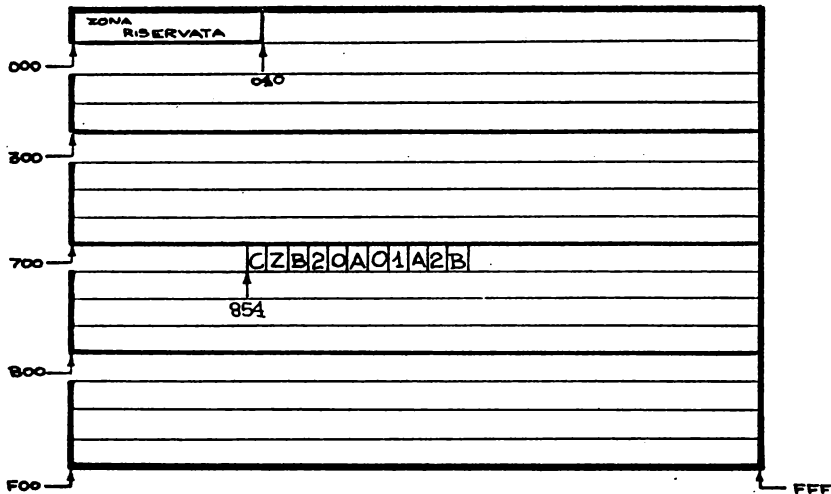
La seguente istruzione :

B	4	0	A	0	8	5	4

provoca la stampa del contenuto alfanumerico di 11 bytes della memoria a partire dall'indirizzo 854 (2132 in decimale).

Se il contenuto della memoria prima dell'esecuzione della istruzione è il seguente :

verrà effettuata la seguente stampa :

CZB20A01A2B

3.3 ISTRUZIONI DI CONTROLLO

Le istruzioni appartenenti a questa classe sono tutte di formato B.

La loro funzione è di alterare l'ordine di esecuzione delle istruzioni di un programma che ordinariamente è quello sequenziale, condizionando eventualmente tale alterazione ad un determinato risultato aritmetico o all'esito di un precedente confronto fra i dati.

Tra le istruzioni di questa classe si trovano anche l'istruzione di arresto, di ovvio significato, e l'istruzione di salto con memorizzazione che trova il suo impiego nel collegamento con i sottoprogrammi (paragrafo 5.2).

Le funzioni delle istruzioni appartenenti a questo gruppo vengono dettagliatamente descritte nel seguito.

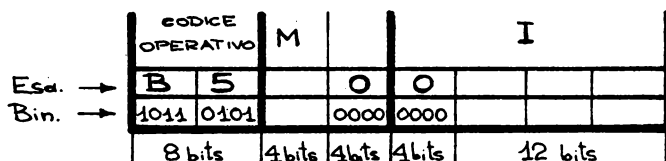
3.3.1 SALTO

Funzione

Il formato di questa istruzione è di tipo B; essa provoca il confronto tra la configurazione di quattro bits specificata in M (maschera) e quella che viene generata automaticamente dal calcolatore, a partire dal codice di condizione, secondo le modalità descritte più avanti.

Se tali configurazioni hanno almeno un bit 1 in comune, l'indirizzo specificato in I viene caricato nel Registro stato del programma al posto dell'indirizzo dell'istruzione successiva e sarà pertanto l'istruzione che si trova a tale indirizzo ad essere successivamente eseguita.

Se le configurazioni non hanno alcun bit 1 in comune l'esecuzione del programma prosegue in modo sequenziale.

	CODICE OPERATIVO		M		I			
Esa. →	B	5		0	0			
Bin. →	1011	0101		0000	0000			
	8 bits		4bits	4bits	4bits	12 bits		

Codice operativo = B5 CODICE SIMBOLICO=SAL

M = maschera di 4 bits (0 ÷ F esadecimale)

I = indirizzo dell'eventuale salto.

Fig. 3.12 - Formato della istruzione di "Salto"

Codice Condizione

Il codice di condizione viene utilizzato da questa istruzione ma non viene da essa alterato.

All'atto della esecuzione della istruzione di salto, il calcolatore genera, a partire dal codice di condizione, una configurazione di quattro bits, che chiameremo "codice di condizione esteso", secondo il seguente schema :

Codice di condizione		Codice di condizione esteso
00	⟶	1000
01	⟶	0100
10	⟶	0010
11	⟶	0001

Maschera

Il valore nella maschera M serve per indicare la condizione il cui verificarsi provoca il salto.

I valori della maschera M assumono diversi significati che dipendono dal tipo di istruzione che ha agito per ultima sul codice di condizione.

Gli unici due valori di M che hanno sempre lo stesso significato sono :

0   il salto non viene effettuato

F   il salto viene sempre effettuato ("salto incondizionato").

Dopo una istruzione aritmetica devono essere utilizzati i seguenti valori di M :

8 viene effettuato il salto se il risultato è uguale a zero

2 viene effettuato il salto se il risultato è maggiore di zero

4 viene effettuato il salto se il risultato è minore di zero

1 viene effettuato il salto se si è verificato overflow

7 viene effettuato il salto se il risultato è diverso da zero

D viene effettuato il salto se il risultato è non positivo

B viene effettuato il salto se il risultato è non negativo

E viene effettuato il salto se non si è verificato overflow

Dopo una istruzione di confronto devono essere utilizzati i seguenti valori :

2 viene effettuato il salto se il 1° operando è maggiore del 2° operando

4 viene effettuato il salto se il 1° operando è minore del 2° operando

8 viene effettuato il salto se il 1° operando è uguale al 2° operando

D viene effettuato il salto se il 1° operando è non maggiore del 2° operando

B viene effettuato il salto se il 1° operando è non minore del 2° operando

7 viene effettuato il salto se il 1° operando è non uguale al 2° operando.

Esempio

La seguente istruzione :

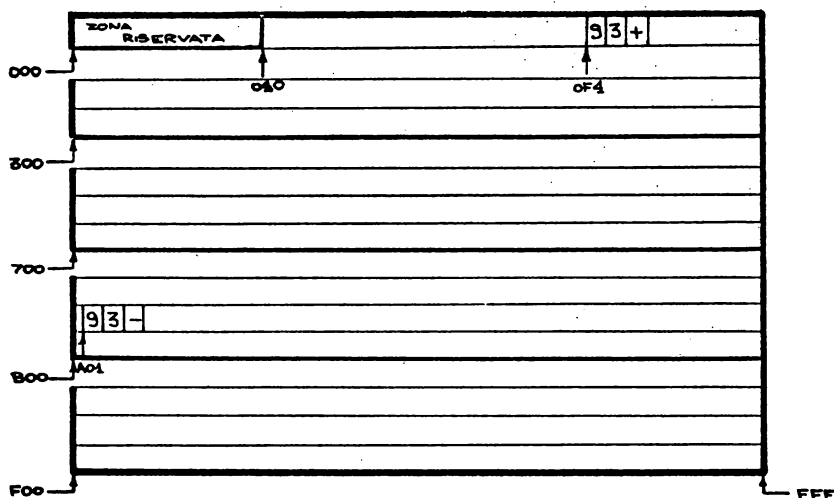
B	5	8	0	0	8	A	C

provoca il salto all'istruzione di indirizzo 8AC (2220 in decimale) se l'ultima istruzione che ha agito sul codice di condi - zione è stata una istruzione aritmetica con il risultato nullo oppure una istruzione di confronto con risultato "uguale".

La maschera infatti ha il valore 8 (1000).

Supponiamo che la precedente istruzione che ha agito sul "codice di condizione" sia stata una "addizione decimale", ad esem - pio :

A1 22 00F4 0A01

Se la mappa della memoria prima dell'esecuzione di tale "addi - zione decimale" è la seguente :

l'istruzione di "salto" sopra considerata provocherà il salto all'indirizzo 8AC in quanto il "codice di condizione" viene impostato dalla istruzione di "addizione decimale" sul valore 0.

3.3.2 SALTO CON MEMORIZZAZIONE

Funzione

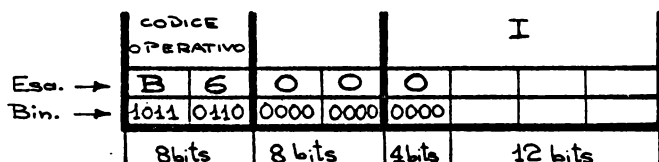
Il formato di questa istruzione è del tipo B.

Essa comanda l'esecuzione dell'istruzione il cui indirizzo è espresso in I, provoca cioè un salto incondizionato all'indirizzo I, e memorizza in un opportuno deposito della zona riservata della memoria e precisamente nei primi due bytes, l'indirizzo della istruzione successiva.

Questa istruzione viene utilizzata per effettuare il richiamo di un "sottoprogramma".

Con essa infatti si risolve facilmente il problema del "rientro" dal sottoprogramma nel "programma principale".

Tale problema viene descritto dettagliatamente nel CAPITOLO V.

	CODICE OPERATIVO				I			
Esa. →	B	6	0	0	0			
Bin. →	1011	0110	0000	0000	0000			
	8bits		8 bits		4bits	12 bits		

Codice operativo = B6 CODICE SIMBOLICO = SME

I = indirizzo dell'istruzione da eseguire.

Fig. 3.13 - Formato del "Salto con memorizzazione"

Codice Condizione

Il "codice di condizione" non viene gestito da questa istruzione.

Esempio

La seguente istruzione :

provoca il "salto incondizionato" all'istruzione di indirizzo 062A, e memorizza in un deposito della "zona riservata" della memoria l'indirizzo dell'istruzione sequenzialmente successiva ad essa nel programma.

3.3.3 ARRESTO

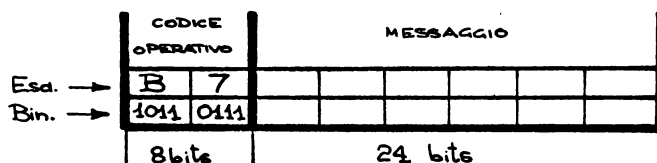
Funzione

Il formato di questa istruzione è del tipo B.

Essa provoca la fine dell'esecuzione di un programma.

Quando viene incontrata questa istruzione, il calcolatore comanda la stampa di un messaggio di avviso, nel quale compare il contenuto esadecimale degli ultimi tre bytes dell'istruzione stessa.

La stampa esadecimale del contenuto degli ultimi tre bytes dell'istruzione permette di identificare su quale istruzione di arresto è terminato il programma; occorre infatti ricordare che in ogni programma possono esistere più istruzioni di arresto.

	CODICE OPERATIVO		MESSAGGIO					
Esa. →	B	7						
Bin. →	1011	0111						
	8 bits		24 bits					

Codice Operativo = B7 CODICE SIMBOLICO = ALT

MESSAGGIO = insieme di caratteri esadecimali che distingue questa particolare istruzione di arresto.

Fig. 3.14 - Formato della istruzione di "Arresto"

Codice Condizione

Il "codice di condizione" non viene gestito da questa istruzione

Stampa di avviso

Quando viene incontrata questa istruzione, automaticamente è stampato il seguente messaggio :

ALT XXXXXX

in cui i 6 X rappresentano il contenuto esadecimale degli ultimi 3 bytes dell'istruzione di arresto.

Esempio

La seguente istruzione :

B	7	A	0	0	1	5	2

da luogo alla stampa del seguente messaggio :

ALT A00152

e pone termine all'esecuzione del programma.

CAPITOLO 4

TECNICHE DI PROGRAMMAZIONE

4.1 ANALISI DI UN PROBLEMA E DIAGRAMMA DI FLUSSO

4.2 STRUTTURA DEI PROGRAMMI IN LINGUAGGIO MACCHINA

4.3 MODIFICA DELLE ISTRUZIONI

4.4 ESEMPI DI PROGRAMMI IN LINGUAGGIO MACCHINA

4.1 ANALISI DI UN PROBLEMA E DIAGRAMMA DI FLUSSO

Nel presente capitolo vengono riportati alcuni esempi di soluzioni di problemi mediante programmi che utilizzano le istruzioni del linguaggio macchina descritte nel capitolo III.

Come è noto, la risoluzione di un problema mediante l'uso del calcolatore può essere considerata come suddivisa in 3 parti, o "fasi", che in ordine di tempo svolgono le seguenti funzioni :

a - FASE I - Analisi del problema e scelta dell'algoritmo da utilizzare

b - FASE II - Stesura del "diagramma di flusso" o "flow-chart" che descrive in forma grafica l'algoritmo scelto nella FASE I

c - FASE III - Codifica e messa a punto del programma

Nella fase di analisi devono essere definite in modo dettagliato tutte le caratteristiche del problema da risolvere (formato dei dati in input, controlli vari di validità, formato dei dati in output, ecc.) e deve essere scelto l'algoritmo realizzante la sequenza operativa richiesta dal problema.

La FASE II prevede la stesura di un "diagramma di flusso" avente lo scopo di visualizzare la sequenza operativa dell'algoritmo scelto nella FASE I. Tale "diagramma di flusso" deve essere utilizzato come base per la successiva codifica delle istruzioni.

Per la stesura di tale diagramma sono fissate delle norme in modo da realizzare un vero e proprio linguaggio che può essere considerato come intermedio tra l'usuale linguaggio umano ed il linguaggio macchina, proprio del calcolatore.

Le convenzioni attualmente in uso prevedono l'uso di alcuni simboli standard;tra quelli di più frequente impiego si trovano quelli riportati nella figura 4.1.

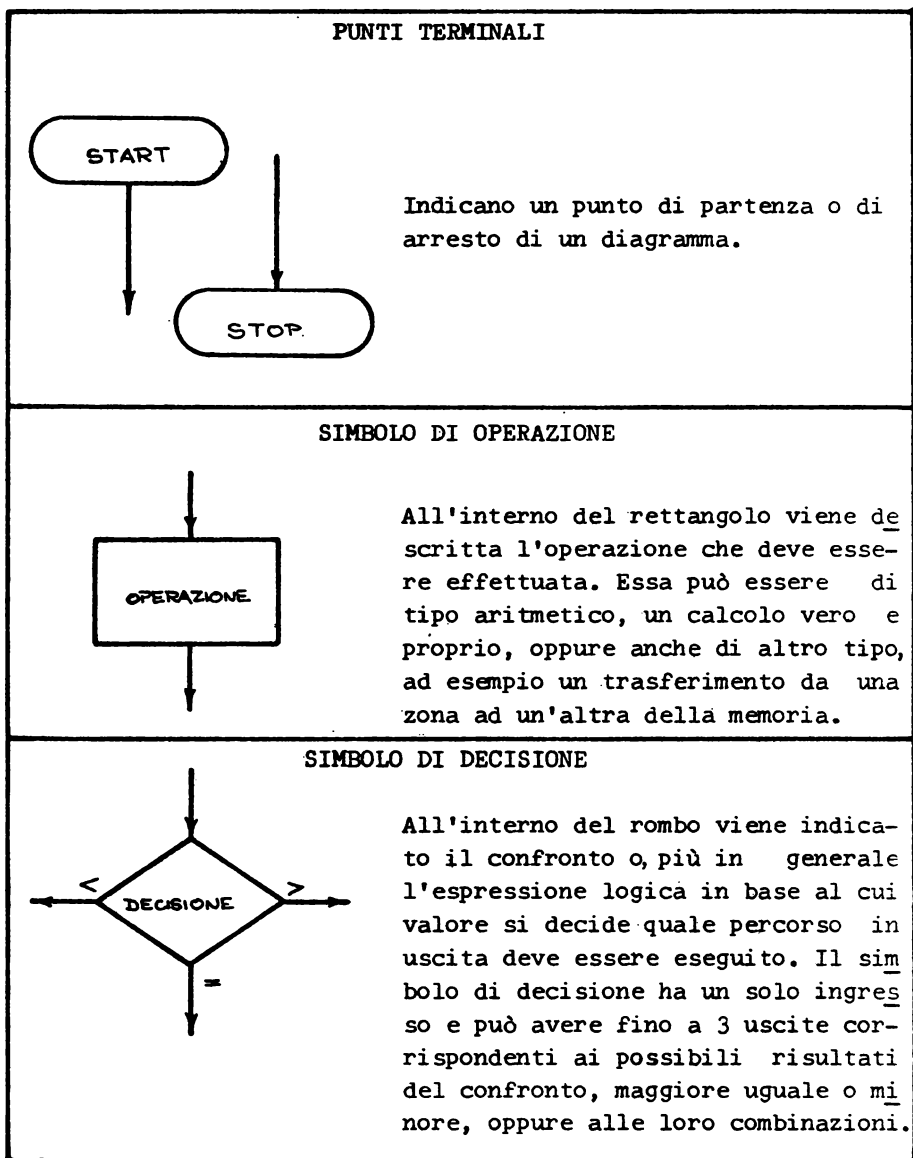
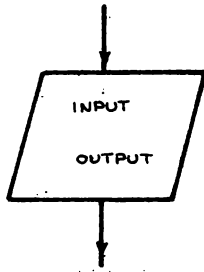



Fig. 4.1 - Simboli standard per flow-charts

SIMBOLO DI INPUT/OUTPUT

Questo simbolo viene utilizzato per indicare l'input o l'output di dati. I dati da introdurre o da estrarre devono essere descritti in dettaglio all'interno del parallelogramma.

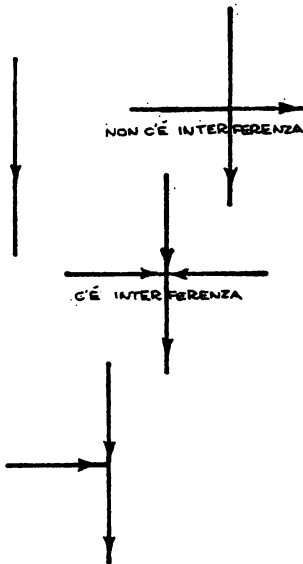
LINEE DI COLLEGAMENTO

Generalmente i simboli di 'operazione', 'decisione', 'input ed output' 'start', 'stop' si susseguono nel diagramma dall'alto verso il basso e sono collegati mediante tratti rettilinei sui quali è stabilito un orientamento, concordemente all'ordine di esecuzione delle istruzioni da essi rappresentate.
Le linee di collegamento possono non essere orientate se questo non dà luogo a possibili equivoci.
La freccia diventa indispensabile per distinguere ad esempio il caso di semplice intersezione di due linee dal loro raggruppamento.

Fig. 4.1/bis - Simboli standard per flow-charts

Si ritiene a questo punto non opportuno dare delle norme per la effettiva stesura dei "diagrammi di flusso" anche perchè dare norme che abbiano una efficacia generale non sarebbe facile.

In ogni caso l'algoritmo risolutivo di tutti i problemi che vengono descritti nel seguito viene anche rappresentato nella forma di flow-chart. Da questi esempi possono pertanto essere tratti degli orientamenti di carattere generale.

Per la redazione del programma (codifica) devono essere eseguite le regole esposte in precedenza nel paragrafo 2.4.

La "messa a punto" o "debugging" di un programma, che insieme alla codifica costituisce la fase III, consiste nell'insieme di prove che devono essere effettuate per eliminare gli errori che sono presenti inizialmente nel programma codificato e per verificare quindi il corretto funzionamento.

4.2 STRUTTURA DEI PROGRAMMI IN LINGUAGGIO MACCHINA

Generalmente al momento della codifica delle varie istruzioni componenti un programma non sono ancora noti gli indirizzi di tutte le costanti ed aree di lavoro utilizzate nel programma.

Pertanto in un primo momento alcune istruzioni non possono essere codificate in modo completo. In esse al posto degli indirizzi incogniti vengono usualmente utilizzati dei riferimenti mnemonici di vario tipo. Il programmatore dovrà tenersi un piccolo elenco con tutti i riferimenti usati, e con le indicazioni necessarie per risalire agli indirizzi che ad essi andranno sostituiti.

Tale procedura può essere schematizzata dalle seguenti due fasi che si svolgono sequenzialmente :

1^ FASE - scrittura dei codici operativi, delle lunghezze degli operandi, dei valori degli indirizzi noti e dei simboli di riferimento sostituenti i valori degli indirizzi conosciuti al momento della scrittura.

2^ FASE - sostituzione dei vari simboli di riferimento utilizzati nella FASE precedente con i valori effettivi degli indirizzi da essi rappresentati.

E' possibile ridurre notevolmente il numero degli indirizzi non conosciuti al momento della scrittura di una istruzione strutturando in modo opportuno le zone di memoria occupate dalle istruzioni, dalle costanti e dalle aree di lavoro del programma.

Una struttura molto semplice, che però presuppone l'esatta conoscenza di tutte le costanti e di tutte le aree di lavoro utilizzate dal programma è indicata nella figura 4.2.

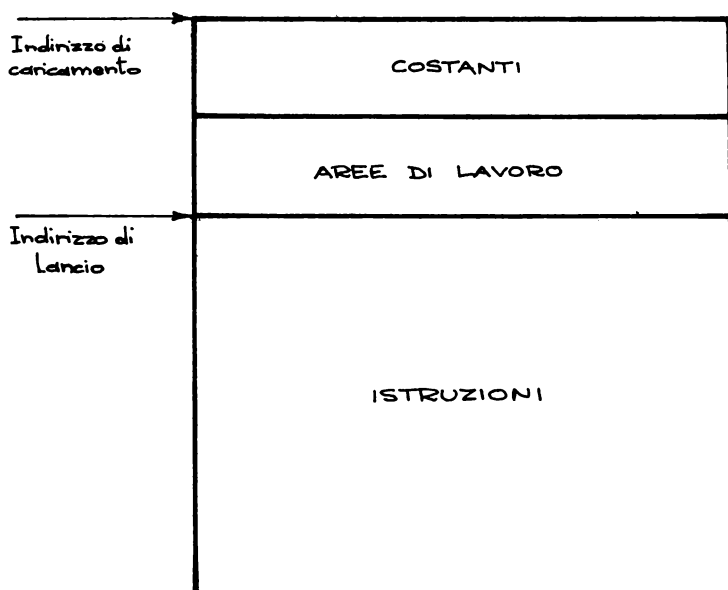



Figura 4.2

In tal modo tutte le istruzioni nelle quali compaiono riferimenti alle costanti ed alle aree di lavoro possono essere codificate in modo completo in quanto tutti gli indirizzi sono già noti in precedenza.

Una struttura che non richiede le ipotesi precedenti, e quindi che è più facilmente applicabile a qualsiasi genere di programma è indicata nella figura 4.3.

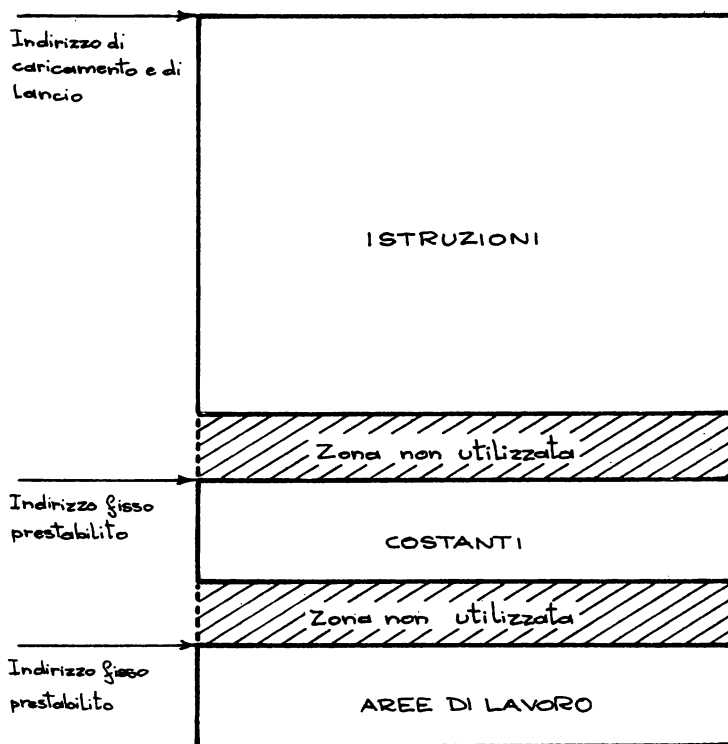



Figura 4.3

In tale struttura tutte le costanti devono essere registrate in una zona di memoria ad esse riservata ed avente un indirizzo iniziale prefissato. Ciò permette di stabilire l'indirizzo di una costante al momento della sua definizione.

Un discorso analogo vale anche per le aree di lavoro (depositi, aree di input, aree di output, ecc) richieste dal programma.

Gli indirizzi di inizio della zona riservata alle costanti e di quella riservata alle aree di lavoro che non sono necessariamente distinte devono essere stabiliti in modo tale che la zona oc-

cupata dal 'testo' del programma (cioè dalle istruzioni) non arrivi mai a sovrapporsi ad esse.

Occorre notare però che un programma organizzato secondo questo tipo di struttura occupa più posizioni di memoria che non se fosse organizzato secondo il tipo di struttura di fig. 4.2.

Infatti sicuramente rimangono delle posizioni di memoria non occupate tra la fine del 'testo' del programma (istruzioni) e l'inizio della zona riservata alle costanti tra la fine di quest'ultima zona e l'inizio della zona riservata alle aree di lavoro.

Un tipo di struttura decisamente più laborioso da realizzare ma che evita sia la compilazione differita degli indirizzi delle costanti e delle aree di lavoro sia uno spreco delle posizioni di memoria è quella della figura 4.4.

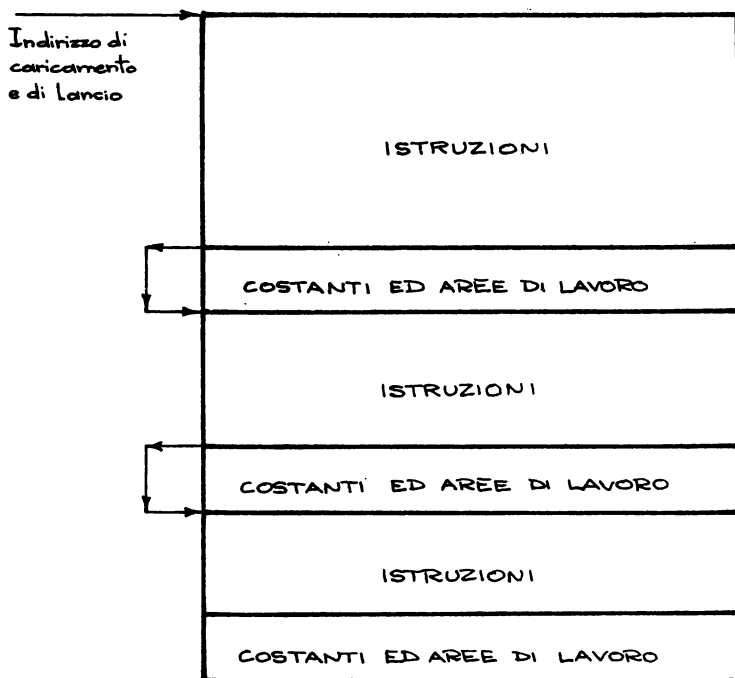



Figura 4.4

In tale tipo di struttura le costanti e le aree di lavoro vengono registrate in memoria in parecchie zone situate all'interno del 'testo' del programma.

L'ampiezza di tali zone è variabile e dovrà essere stabilita dal programmatore tenendo presente che esse devono sicuramente essere riempite prima della fine del programma.

Un metodo pratico che deve essere seguito per realizzare una struttura di questo tipo è quello di generare una nuova zona riservata alle costanti ed alle aree di lavoro soltanto quando la precedente zona riservata è piena.

Ciascuna zona riservata dovrà essere preceduta da una istruzione di salto incondizionato all'indirizzo della istruzione successiva alla fine della zona riservata, per evitare che il suo contenuto venga considerato come 'testo' del programma, e quindi che venga 'eseguito' anch'esso.

4.3 MODIFICA DELLE ISTRUZIONI

Nella risoluzione dei problemi mediante il calcolatore capita molto frequentemente di dovere effettuare uno stesso gruppo di operazioni su un insieme di dati registrati consecutivamente nella memoria.

Sono esempi banali di tali applicazioni la ricerca dell'elemento dell'insieme avente un particolare valore, oppure avente il valore massimo, o minimo, ecc. l'aggiornamento di un elemento in una particolare posizione, oppure la somma di un determinato numero di elementi, ecc. ecc.

Per la risoluzione degli esempi ora citati devono essere effettuati su tutti gli elementi dell'insieme confronti, somme o altre operazioni elementari. E' chiaro che se nei relativi programmi si ripete lo stesso gruppo di istruzione per ogni dato dell'insieme, i programmi stessi risulterebbero estremamente lunghi e legati al numero degli elementi contenuti nell'insieme.

Per riuscire a realizzare programmi più concisi e che non dipendano dal numero degli elementi dell'insieme considerato si utilizza il metodo della modifica degli indirizzi.

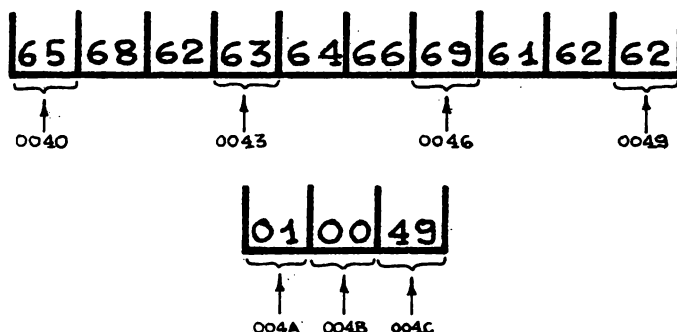
Si consideri ad esempio il problema di effettuare la somma decimale di dieci elementi di un insieme aventi i seguenti valori:

5, 8, 2, 3, 4, 6, 9, 1, 2, 2

e registrati nella memoria del calcolatore a partire dall'indirizzo 0040.
Il risultato di tale somma deve essere registrato in un deposito lungo 2 bytes di indirizzo 0050 preventivamente azzerato.

Si suppone che nel byte di indirizzo 004A sia memorizzata la costante esadecimale 01 (lunghezza di ogni dato dell'insieme) e che nei due bytes successivi sia memorizzato il valore del

l'indirizzo dell'ultimo elemento.

Nella memoria si trovano quindi i seguenti dati:

Le istruzioni costituenti la seguente parte di programma:

INDIRIZZO DI MEMORIA	ISTRUZIONI	
....	.. ..	
0A52	A1 10 0050 0040	(ADE)
0A58	A3 31 0A56 004A	(AES)
....	.. ..	
....	.. ..	

provocano nell'ordine le seguenti operazioni:

1° - somma decimale dell'elemento dell'insieme il cui indirizzo è espresso nel 2° operando con il contenuto del deposito di indirizzo 0050

2° - somma esadecimale della costante 01 all'indirizzo dell'elemento dell'insieme considerato nella istruzione precedente. Il risultato di tale somma è l'indirizzo del dato

successivo e viene posto nel campo secondo operando della istruzione precedente.

In tal modo l'indirizzo del secondo operando della istruzione di somma decimale viene modificato automaticamente dalla istruzione di somma esadecimale.

Eseguendo le precedenti istruzioni tante volte quanti sono gli elementi dell'insieme si otterrà il valore della somma direttamente nel deposito di indirizzo 0050.
Naturalmente tali istruzioni devono essere seguite da una 'istruzione di confronto' dell'indirizzo del dato considerato con l'indirizzo dell'ultimo dato dell'insieme e da una 'istruzione di salto' condizionato.

Queste ultime servono per stabilire quando sono stati sommati tutti gli elementi e per rimandare il calcolatore ad eseguire nuovamente l'istruzione di somma decimale, quando non sono ancora stati sommati tutti i dati.

Questo metodo di modifica è piuttosto laborioso, in ogni caso è l'unico possibile per realizzare i loops nei programmi che utilizzano le istruzioni del linguaggio macchina descritto in precedenza: nel cap. VI si vedrà che con l'introduzione dei 'registri indice' nel calcolatore CND tale aggiornamento verrà molto facilitato.

4.4 ESEMPI DI PROGRAMMI IN LINGUAGGIO MACCHINA

Tutti i problemi considerati nel seguito vengono esposti secondo il seguente schema:

1 - TESTO del problema (comprendente anche la eventuale analisi)

2 - FLOW-CHART

3 - PROGRAMMA realizzato in linguaggio macchina seguito da una descrizione della struttura del programma e delle tecniche usate nella codifica.

4.4.1 ESEMPIO 1

Testo

Costruire un programma che consenta la introduzione di 35 numeri decimali non segnati da due cifre.

Per ogni numero introdotto dovranno essere effettuate le seguenti operazioni:

a – stabilire se il suo valore è maggiore o minore di 50

b – se il suo valore è maggiore od uguale a 50 tale numero dovrà essere sommato al contenuto di un deposito DEP1, in caso contrario dovrà essere sommato al contenuto di un deposito DEP2.

Durante tutto il procedimento di calcolo nella memoria del calcolatore deve essere presente uno soltanto dei numeri introdotti.
Alla fine del programma dovranno essere stampati i valori decimali dei due depositi.

Analisi

Ogni numero introdotto viene registrato in una stessa zona di memoria, la quale svolge soltanto la funzione di 'zona di transito' per i dati. Infatti, dopo aver stabilito se il valore del numero in essa contenuto è maggiore o minore di 50, esso verrà prelevato da tale zona e sommato al deposito interessato.

Flow-chart

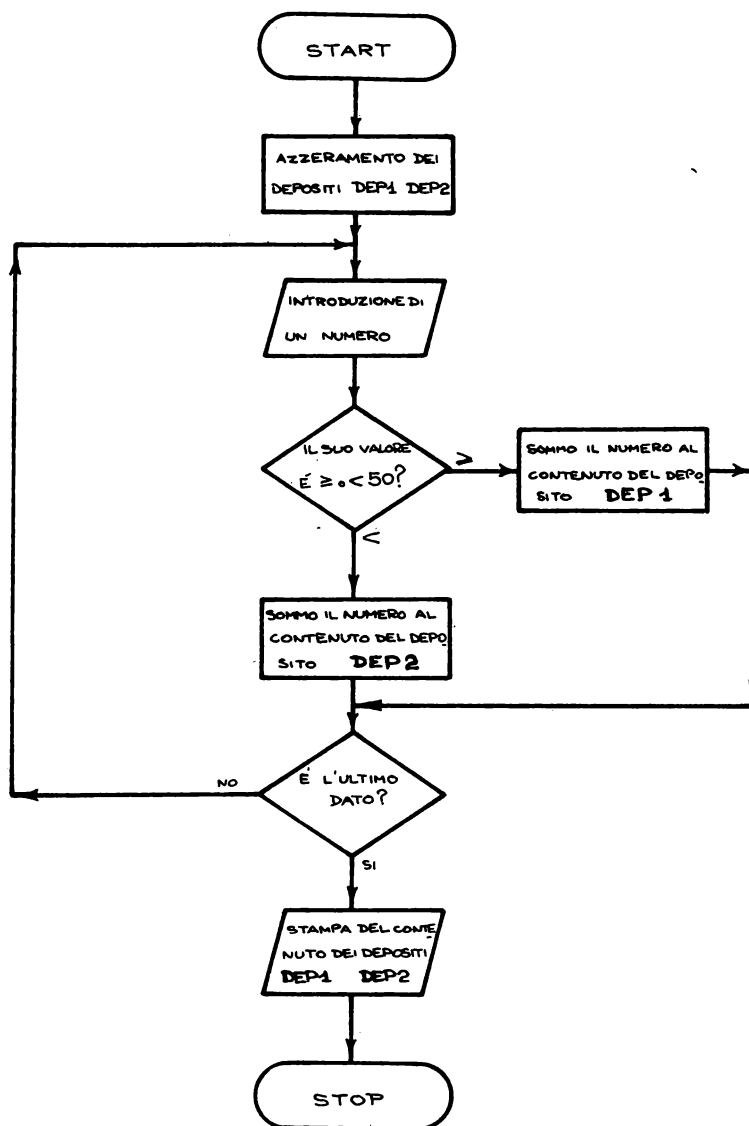



Figura 4.5

Programma

Le costanti e le aree di lavoro che servono per la realizzazione del programma sono le seguenti:

2 bytes	per contenere il numero introdotto (zona di transito)
2 bytes	per contenere la costante decimale '50' (da confrontare con ogni numero)
4 bytes	per contenere la somma decimale dei numeri di valore $\geqslant 50$ (deposito DEP1)
4 bytes	per contenere la somma decimale dei numeri di valore < 50 (deposito DEP2)
1 byte	per contenere la costante decimale '1' che serve per contare il numero di dati introdotti
2 bytes	per contenere il numero dei dati già introdotti
2 bytes	per contenere la costante decimale '35', che serve per arrestare il procedimento di calcolo dopo la introduzione del 35° numero.

Per esse viene riservata una zona di memoria lunga 17 bytes, arrotondati a 18, (12 in esadecimale) a partire dalla posizione di indirizzo 0040 (prima posizione della memoria utilizzabile).

I valori iniziali dei depositi (generalmente azzerati) e quelli delle costanti vengono riportati su un 'foglio di programmazione' utilizzato in questo caso come mappa della memoria; in tale modo tutti i loro indirizzi sono già noti al momento della codifica delle istruzioni del programma.

Le istruzioni del linguaggio macchina costituenti il programma devono essere registrate in memoria a partire dall'indirizzo 0052 (0040+12), e da tale indirizzo devono pure essere eseguite.

Il 'foglio di programmazione' contenente i valori iniziali dei depositi e quelli delle costanti è illustrato nella figura 4.6, mentre il 'foglio di programmazione' contenente il programma è illustrato nella figura 4.7.

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	0	0	4	0				6	0	6	0									DEPOSITO DI TRANSITO (2 bytes)
	0	0	4	2				6	5	6	0									COSTANTE DECIMALE '50'
	0	0	4	4				6	0	6	0	6	0	6	0					DEPOSITO DEP1
	0	0	4	8				6	0	6	0	6	0	6	0					DEPOSITO DEP2
	0	0	4	C				6	1											COSTANTE DECIMALE '1'
	0	0	4	D				6	0	6	0									DEPOSITO PER IL NUMERO DEI DATI INTRODOTTI
	0	0	4	F				6	3	6	5									COSTANTE DECIMALE '35
	0	0	5	1				6	0											CARATTERE RIEMPITIVO

Fig. 4.6 - Esempio 1 - Costanti e Depositi

SALTI	INDIRIZZO ISTRUZIONE (1-4)	CODICE SIMBOL. (5-7)	ISTRUZIONE: COD. OPER. (8-9)	LUNG. (10-11)	1° OPERANDO (12-15)	2° OPERANDO (16-19)	OSSERVAZIONI
	0052	SDE	A2	77	0044	0044	AZZERAMENTO DEP1 DEP2
	0058	IAL	B2	01	0040		INTRODUZIONE DI DUE CARATTERI NUMERICI
	005C	CAL	A7	11	0040	0042	CONFRONTO DEL NUMERO INTRODOTTO CON '50'
	0062	SAL	B5	B0	0070		SE IL RISULTATO É ≥ SALTO A INDIRIZZO 0070
	0066	ADE	A1	31	0048	0040	SOMMA DEL NUMERO AL DEPOSITO DEP2
	006C	SAL	B5	F0	0076		SALTO INCONDIZIONATO ALL'INDIRIZZO 0076
	0070	ADE	A1	31	0044	0040	SOMMA DEL NUMERO AL DEPOSITO DEP1
	0076	ADE	A1	10	004D	004C	SOMMA '1' AL DEPOSITO 'NUMERO DATI INTRODOTTI'
	007C	CAL	A7	11	004D	004F	CONFRONTO DEPOSITO 'NUMERO DATI INTRODOTTI' CON '35'
	0082	SAL	B5	D0	0058		SE IL RISULTATO É ≤ SALTO A INDIRIZZO 0058
	0086	OAL	B4	03	0044		STAMPA DEL CONTENUTO DI DEP1
	008A	OAL	B4	03	0048		STAMPA DEL CONTENUTO DI DEP2
	008E	ALT	B7	00	0001		FINE DEL PROGRAMMA

Fig. 4.7 - Esempio 1 - Programma

Le funzioni svolte dalle singole istruzioni durante l'esecuzione del programma sono le seguenti:

Sott. dec.	- SDE -	Provoca l'azzeramento dei depositi DEP1 e DEP2
Intr. alf.	- IAL -	Sblocca la tastiera della telescrivente consentendo l'introduzione di 2 caratteri alfanumerici (2 cifre costituenti il numero da elaborare)
Confr. alg.	- CAL -	Provoca il confronto algebrico del numero introdotto con '50'
Salto	- SAL -	Se il risultato del precedente confronto è $\geq$ il numero deve essere sommato al deposito DEP1, e quindi deve essere eseguita successivamente l'istruzione di indirizzo 0070
Add. dec.	- ADE -	Provoca la somma decimale del numero al deposito DEP2
Salto	- SAL -	Se viene eseguita l'istruzione ADE precedente (numero di valore $<$ di 50) l'istruzione ADE successiva (somma del numero al deposito DEP1) non dovrà essere eseguita. Pertanto questa istruzione è di salto incondizionato all'indirizzo 0076
Add. dec.	- ADE -	Provoca la somma decimale del numero al deposito DEP1
Add. dec.	- ADE -	Provoca la somma decimale di 1 al deposito che contiene il numero dei dati introdotti e quindi già elaborati
Confr. alg.	- CAL -	Provoca il confronto del deposito che contiene il numero dei dati introdotti con il numero '35'
Salto	- SAL -	Se il risultato del precedente confronto è $\leq$ si salta ad eseguire nuovamente il programma dall'inizio (istruzione IAL), in caso contrario viene eseguita l'istruzione OAL successiva

Estr. alf.	- OAL -	Provoca la stampa alfanumerica (decimale) del contenuto del deposito DEP1
Estr. alf.	- OAL -	Provoca la stampa alfanumerica (decimale) del contenuto del deposito DEP2
Arresto	- ALT -	Provoca la fine della elaborazione e la stampa del seguente messaggio: ALT 000001

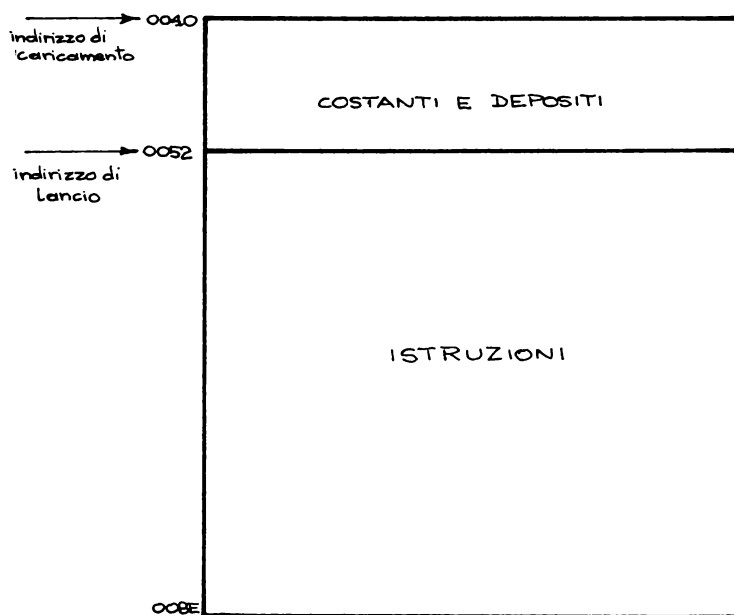
L'esecuzione di un programma con i seguenti dati:

15,82,61,12,10,33,66,25,86,91
65,41,14,61,16,68,33,46,19,50
24,29,81,96,99,58,18,49,48,40
42,82,32,62,26

provoca la seguente stampa:

```
 15
 82
 61
 12
 10
 33
 66
 25
 86
 91
 65
 41
 14
 61
 16
 68
 33
 46
 19
 50
 24
 29
 81
 96
 99
 58
 18
 49
 48
 40
 42
 82
 32
 62
 26
 1108
  572

                          ALT 000001
```

La struttura adottata per tale programma è la seguente:

L'occupazione di memoria del programma è di 0051 bytes (81 in decimale).

4.4.2 ESEMPIO 2

Testo

Costruire un programma che consenta la introduzione consecutiva di 10 numeri decimali non segnati di 2 cifre.
Alla fine di tale introduzione dovrà essere calcolata e stampata la loro somma.

Analisi

Per risolvere tale problema, a differenza del precedente, si caricano contemporaneamente in memoria tutti i dati. Si realizza quindi in memoria una successione di 10 elementi; ciascun elemento occupa due bytes (i numeri infatti sono di 2 cifre).

Ogni elemento della successione viene sommato al contenuto di un deposito che, alla fine del procedimento, conterrà il valore della somma.

Gli indirizzi degli elementi della successione possono essere calcolati in modo iterativo a partire dal primo sommando la costante '02' (lunghezza in bytes di ogni elemento) all'indirizzo dell'elemento precedente.

Per la realizzazione di questi calcoli si ricorre al procedimento di modifica delle istruzioni descritto nel paragrafo 4.3.

Flow-chart

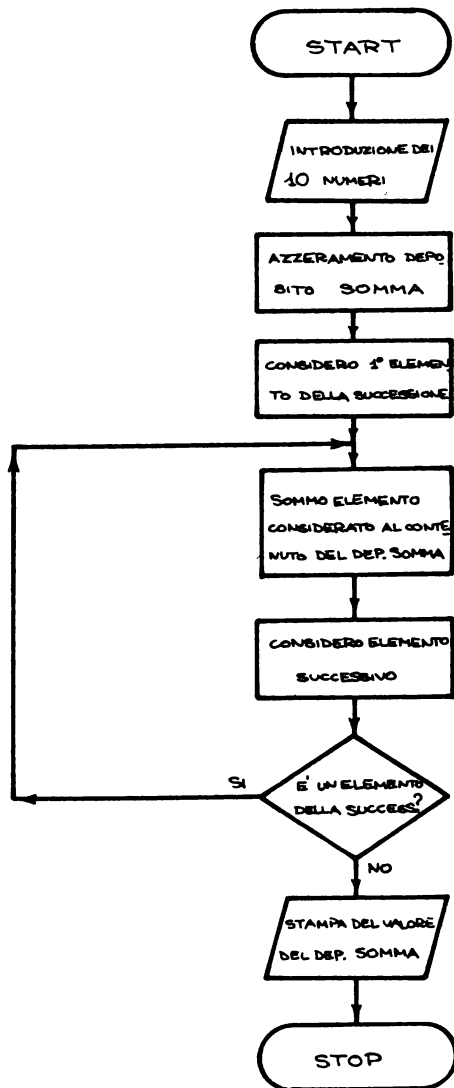



Figura 4.8

Programma

Data la semplicità del problema considerato è facilmente determinabile il numero esatto dei bytes occupati dalle costanti e dalle aree di lavoro (depositi) del programma.

Essi sono:

20 bytes	che contengono i 10 numeri dati
3 bytes	che contengono il risultato della somma
1 byte	che contiene il valore esadecimale '02', utilizzato per incrementare gli indirizzi degli elementi della successione
2 bytes	che contengono il valore esadecimale dell'indirizzo dell'ultimo elemento della successione

Per esse viene quindi riservata una zona di memoria lunga 26 bytes (001A esadecimale) di indirizzo 0040.
I valori delle costanti e dei depositi vengono riportati su un 'foglio di programmazione' utilizzato unicamente come mappa delle memoria; in tale modo tutti i loro indirizzi sono già noti al momento della codifica delle istruzioni del programma.

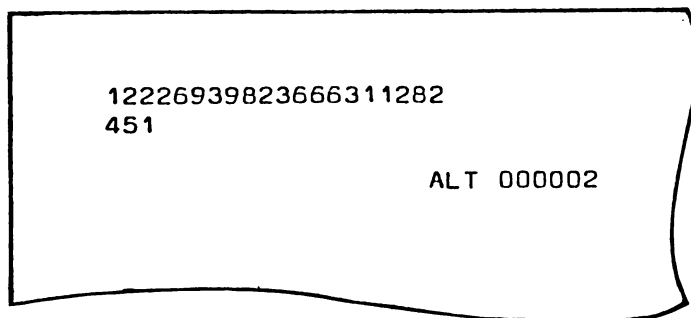
Le istruzioni del linguaggio macchina costituenti il programma vengono quindi registrate in memoria a partire dall'indirizzo 005A (ottenuto come risultato della somma 0040 + 001A), e da tale indirizzo devono pure essere eseguite.

Il 'foglio di programmazione' contenente i valori delle costanti e dei depositi è illustrato nella figura 4.9, mentre il 'foglio di programmazione' contenente il programma è illustrato nella figura 4.10.

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	0	0	4	0				6	0	6	0	6	0	6	0	6	0	6	0	
	0	0	4	6				6	0	6	0	6	0	6	0	6	0	6	0	AREA PER LA REGISTRAZIONE DEI 10 NUMERI
	0	0	4	C				6	0	6	0	6	0	6	0	6	0	6	0	DATI
	0	0	5	2				6	0	6	0									
	0	0	5	4				6	0	6	0	6	0							DEPOSITO PER IL VALORE DELLA SOMMA
	0	0	5	7				0	2											COSTANTE ESADECIMALE '02'
	0	0	5	8				0	0	5	2									INDIRIZZO ULTIMO ELEMENTO DA SOMMARE

Fig. 4.9 - Esempio 2 - Costanti e Depositi

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	0	0	5	A	I	A	L	B	2	1	3	0	0	4	0					INTRODUZIONE DI 20 CARATTERI ALFANUMERICI
	0	0	5	E	S	D	E	A	2	2	2	0	0	5	4	0	0	5	4	AZZERAMENTO DEPOSITO SOMMA
	0	0	6	4	A	D	E	A	1	2	1	0	0	5	4	0	0	4	0	SOMMA DECIMALE ELEMENTO CONSIDER. CON DEP. SOMMA
	0	0	6	A	A	E	S	A	3	3	1	0	0	6	8	0	0	5	7	INCREMENTO INDIRIZZO ELEMENTO CONSIDERATO DI '02'
	0	0	7	0	C	L	O	A	6	0	1	0	0	6	8	0	0	5	8	L'ELEMENTO CONSIDERATO É L'ULTIMO? (INDIR. ≥0052?)
	0	0	7	6	S	A	L	B	5	D	0	0	0	6	4					SE L'INDIRIZZO É ≤ 0052 SALTA ALL'INDIRIZZO 0064
	0	0	7	A	O	A	L	B	4	0	2	0	0	5	4					STAMPA DEL VALORE DEL DEPOSITO SOMMA
	0	0	7	E	A	L	T	B	7	0	0	0	0	0	2					FINE DEL PROGRAMMA

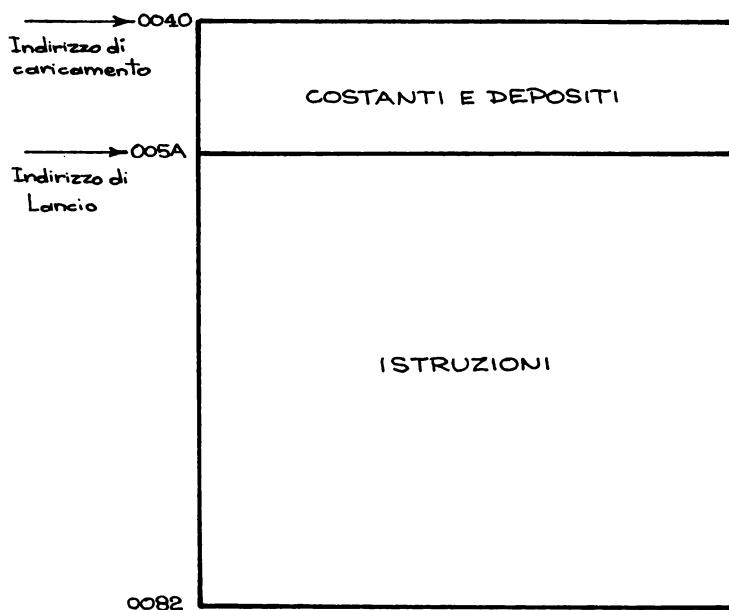
Fig. 4.10 - Esempio 2 - Programma

Le funzioni svolte dalle singole istruzioni durante l'esecuzione del programma sono le seguenti:

Int. alfan.	- IAL -	sblocca la tastiera della telescrivente consentendo l'introduzione di 20 caratteri alfanumerici (10 numeri di 2 cifre). Viene così creata una successione di 10 elementi, il cui primo elemento ha indirizzo '0040'.
Sott. dec.	- SDE -	Provoca l'azzeramento del deposito SOMMA.
Add. dec.	- ADE -	Provoca la somma decimale dell'elemento il cui indirizzo è quello che compare come 2° operando con il contenuto del deposito SOMMA di indirizzo 0054.
Add. esadec.	- AES -	Provoca la somma esadecimale della costante '02' all'indirizzo che compare come 2° operando nell'istruzione precedente. In tal modo si può utilizzare la stessa istruzione ADE per sommare tutti gli elementi al deposito SOMMA.
Compar. log.	- CLO -	Provoca il confronto dell'indirizzo calcolato dalla istruzione precedente con l'indirizzo dell'ultimo elemento (0052).
Salto	- SAL -	Se il risultato del confronto precedente è $\leq$ si salta ad eseguire nuovamente l'istruzione ADE; in caso contrario viene eseguita l'istruzione successiva.
Estraz. alf.	- OAL -	Provoca la stampa alfanumerica del risultato della somma che si trova nel deposito SOMMA di indirizzo 0054.
Arresto	- ALT -	Provoca la fine della elaborazione e la stampa del seguente messaggio: ALT 000002

Si immagini di eseguire il programma introducendo i seguenti numeri:

12 , 22 , 69 , 39 , 82 , 36 , 66 , 31 , 12 , 82

La stampa ottenuta alla fine della esecuzione è la seguente:

La struttura adottata per tale programma è la seguente:

In seguito all'utilizzo di tale struttura l'occupazione di memoria del programma è di 0042 bytes (66 in decimale).

4.4.3 ESEMPIO 3

Testo

Costruire un programma che realizzi nell'ordine le seguenti operazioni:

a – introduzione di 20 numeri decimali segnati di due cifre

b – ricerca e stampa del numero di valore massimo introdotto al punto a

Analisi

Il problema richiede nel punto a la presenza in memoria contemporanea di tutti i 20 dati. Si realizza quindi in memoria una successione di 20 elementi, ciascun elemento occupa 3 bytes (2 caratteri numerici ed 1 per il segno).

Il primo elemento della successione viene posto in un deposito MAX e tutti gli altri elementi vengono confrontati con il contenuto di MAX.

Se l'elemento confrontato è maggiore dell'elemento contenuto nel deposito MAX il suo valore viene sostituito a quello di MAX; in caso contrario si confronta l'elemento successivo.

Quando tutti gli elementi della successione sono stati confrontati l'elemento di massimo valore relativo si troverà nel deposito.

Flow-chart

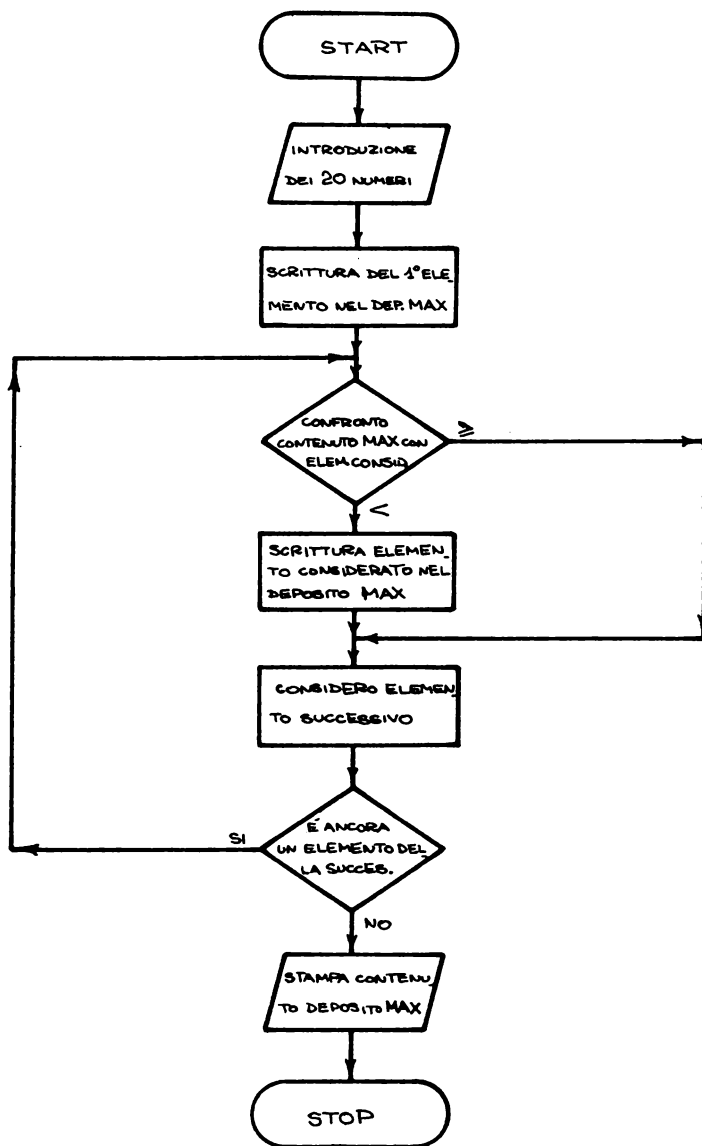



Figura 4.11

Programma

Anche in questo problema è possibile determinare esattamente il numero dei bytes occupati dalle costanti e dai depositi prima di codificare il programma.

Essi sono:

60 bytes	che contengono i 20 numeri dati
1 byte	che contiene il valore esadecimale '03' utilizzato per incrementare i valori degli indirizzi dei dati
2 bytes	che contengono il valore esadecimale dell'indirizzo dell'ultimo dato da considerare
3 bytes	che contengono il numero di massimo valore relativo cercato (deposito MAX)

Per essi viene riservata in memoria una zona lunga 66 bytes (0042 in esadecimale) di indirizzo iniziale 0040.

Gli indirizzi delle costanti e dei depositi possono quindi essere subito calcolati riportando i loro valori su un 'foglio di programmazione', utilizzato in questo caso unicamente come mappa della memoria.

Le istruzioni del linguaggio macchina costituenti il programma vengono quindi registrate in memoria a partire dall'indirizzo 0082 (ottenuto come risultato della somma 0040 + 0042), e da tale indirizzo devono pure essere eseguite.

Il 'foglio di programmazione' contenente i valori delle costanti e dei depositi è illustrato nella figura 4.12, mentre il 'foglio di programmazione' contenente il programma è illustrato nella figura 4.13.

SALTI	INDIRIZZO ISTRUZIONE (1-4)	CODICE SIMBOL. (5-7)	COD. OPER. (8-9)	LUNG. (10-11)	1° OPERANDO (12-15)	2° OPERANDO (16-19)	OSSERVAZIONI
	0040						AREA DI 60 BYTES DESTINATA A CONTENERE
	0046						I 20 NUMERI DA ELABORARE
	004C						
	0052						
	0058						
	005E						
	0064						
	006A						
	0070						
	0076						
	007C		03				COSTANTE ESADECIMALE '03'
	007D		00	79			INDIRIZZO DELL'ULTIMO ELEMENTO
	007F		60	60	60		VALORE INIZIALE DEL DEPOSITO MAX

Fig. 4.12 - Esempio 3 - Costanti e Depositi

SALTI	INDIRIZZO ISTRUZIONE	CODICE SIMBOL.	ISTRUZIONE COD. OPER.	LUNG.	1° OPERANDO	2° OPERANDO	OSSERVAZIONI
	1 2 3 4	5 6 7	8 9	10 11	12 13 14 15	16 17 18 19	
	0082	IAL	B2	3B	0040		INTRODUZIONE DI 60 CARATTERI ALFANUMERICI (20 DATI)
	0086	TRA	A5	02	007F	0040	SCRITTURA DEL 1° ELEMENTO DATO NEL DEPOSITO MAX
	008C	CAL	A7	22	007F	0043	CONFRONTO DEL CONTENUTO DEL DEP. MAX CON ELEM. SUCCES.
	0092	SAL	B5	B0	009C		SE RISULTATO CONFRONTO É ≥ SALTO A INDIRIZZO 009C
	0096	TRA	A5	02	007F	0043	SCRITTURA ELEMENTO CONSIDERATO NEL DEPOSITO MAX
	009C	AES	A3	31	0090	007C	AGGIORNAMENTO INDIRIZZO NELL'ISTRUZIONE DI CONFRONTO
	00A2	AES	A3	31	009A	007C	AGGIORNAMENTO INDIRIZZO NELL'ISTRUZIONE DI TRASFERIM.
	00A8	CLO	A6	01	0090	007D	CONFRONTO INDIRIZZO ELEM. CONSID. CON INDIRIZZO ULTIMO ELEM.
	00AE	SAL	B5	D0	008C		SE RISULTATO CONFRONTO É ≤ SALTO A INDIRIZZO 008C
	00B2	OAL	B4	02	007F		STAMPA VALORE DECIMALE CONTENUTO NEL DEPOSITO MAX
	00B6	ALT	B7	00	0003		FINE DEL PROGRAMMA

Fig. 4.13 - Esempio 3 - Programma

Le funzioni svolte dalle singole istruzioni durante l'esecuzione del programma sono le seguenti:

Intr. alf. - IAL - Sblocca la tastiera della telescrivente consentendo l'introduzione di 60 caratteri alfanumerici costituenti i 20 numeri da elaborare (ogni numero è costituito da 2 caratteri numerici e da 1 per il segno)

Trasferimento- TRA - Provoca il trasferimento del primo elemento nel deposito MAX di indirizzo 007F

Confr. algeb.- CAL - Provoca il confronto algebrico dell'elemento contenuto nel deposito con l'elemento successivo

Salto - SAL - Se il risultato del confronto precedente è ≥ si salta ad eseguire l'istruzione di indirizzo 0096; in caso contrario viene eseguita l'istruzione successiva TRA.

Trasferimento- TRA - Provoca il trasferimento dell'elemento considerato nel deposito MAX di indirizzo 007F

Add. esadec. - AES - Provoca la somma esadecimale della costante '03' all'indirizzo che compare come 2° operando nell'istruzione di 'confronto algebrico'. In tal modo è possibile confrontare gli elementi della successione con il deposito mediante una stessa istruzione

Add. esadec. - AES - Provoca la somma esadecimale della costante '03' all'indirizzo che compare come 2° operando nella precedente istruzione di 'trasferimento'

Confr. log. - CLO - Provoca il confronto logico dell'indirizzo calcolato nelle precedenti istruzioni AES con l'indirizzo dell'ultimo elemento (0079)

Salto - SAL - Se il risultato del precedente confronto è $\leq$ si salta ad eseguire nuovamente l'istruzione di confronto algebrico 'CAL' di indirizzo 008C; in caso contrario viene eseguita l'istruzione successiva OAL

Estraz.Alf. - OAL - Provoca la stampa alfanumerica dell'elemento contenuto nel deposito di indirizzo 007F. Tale elemento sarà quello di massimo valore relativo cercato.

Arresto - ALT - Provoca la fine della elaborazione e la stampa del seguente messaggio:

ALT 000003

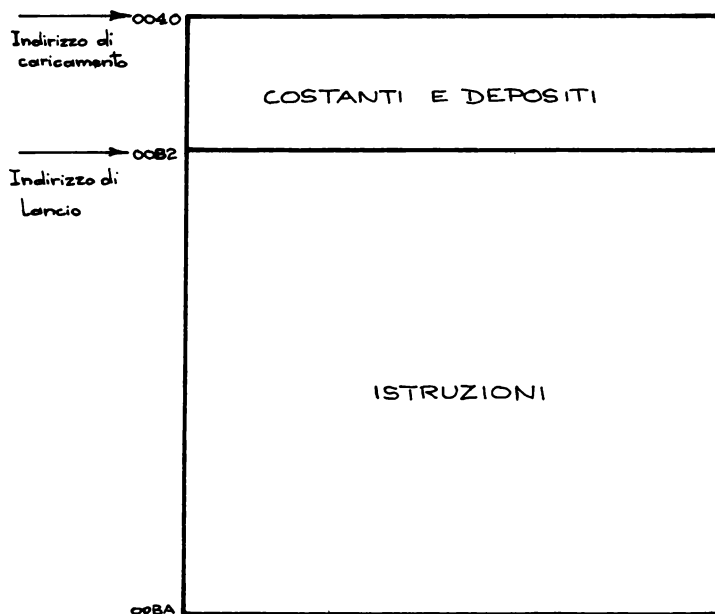
L'esecuzione di questo programma con i seguenti dati numerici:

18+, 90-, 83+, 88+, 22+, 31-, 69-, 96+, 81-, 26-
35-, 54-, 48-, 65+, 81+, 22+, 63-, 24+, 52+, 55+

e con i valori delle costanti e dei depositi riportati nella figura 4.12, da luogo alla seguente stampa:

```
18+90-83+88+22+31-69-96+81-26-35-54-48-65+81+22+63-24+52+55+
96+

                         ALT 000003
```

La struttura adottata per tale programma è la seguente:

L'occupazione di memoria del programma è quindi di 007A bytes, e cioè di 122 caratteri.

4.4.4 ESEMPIO 4

Testo

Costruire un programma che effettui nell'ordine le seguenti operazioni:

a – introduzione di 20 numeri decimali segnati da 2 cifre

b – ordinamento di tali numeri in modo da formare una successione non decrescente

c – stampa degli elementi ordinati e della scritta iniziale secondo il seguente prospetto:

e cioè gli elementi vengono stampati da sinistra verso destra di ogni riga in ragione di 5 per riga.

Analisi

Per poter effettuare l'ordinamento richiesto dal problema nel punto b è necessario avere contemporaneamente in memoria i dati introdotti al punto a che costituiscono una successione di 20 elementi lunghi 3 bytes ciascuno (2 caratteri numerici ed 1 per il segno).

Per realizzare l'ordinamento si ricerca l'elemento di valore minimo tra i 20 elementi dati e lo si scambia con il primo elemento, quindi si ricerca l'elemento di valore minimo tra i rimanenti 19 elementi e lo si scambia con il secondo elemento, e così via sino a che non viene formata la successione non decrescente di 20 elementi voluta.

Per stampare gli elementi ordinati della successione secondo il prospetto esposto nel punto c la riga di stampa deve essere composta in un'area della memoria, e quindi deve essere dato il comando di stampa di tale area.

La lunghezza dell'area di memoria dipende dalla quantità di dati che devono essere stampati; nel problema considerato è necessaria un'area di memoria lunga 24 bytes.

Il deposito lungo 3 bytes necessario per lo scambio dei valori di due elementi della successione durante l'ordinamento viene chiamato SCAMBIO; l'area di memoria lunga 24 bytes necessaria per la stampa viene chiamata STAMPA.

Per poter effettuare l'ordinamento sono necessari altri due depositi lunghi 2 bytes, chiamati rispettivamente DEP1, DEP2, destinati a contenere valori di indirizzi degli elementi della successione.

Flow-chart

START
INTRODUZIONE DI 20 NUMERI
TRASFERIMENTO INDIR. 1° ELEMENTO NEL DEPOSITO DEP1
INDIRIZZO CONTENUTO IN DEP1 INCREMENTATO DI 3 NEL DEPOSITO DEP2
ELEM. DI INDIR. DEP1 ≷ ELEM. DI INDIR. DEP2
>
TRASF. ELEMENTO DI INDIR. CONTENUTO IN DEP1 NEL DEPOSITO SCAMBIO
TRASF. ELEM. INDIR. DEP2 AL POSTO DELL'ELEMENTO DI INDIRIZZO DEP1
≤
TRASF. ELEMENTO CONTENUTO IN SCAMBIO AL POSTO DELL'ELEM. DI INDIRIZZO CONTEN. IN DEP2
INCREMENTO INDIRIZZO CONTENUTO IN DEP2 DI '03'
≤
INDIR. CONTENUTO IN DEP2 ≷ INDIR. ULTIMO ELEMENTO
>
INCREMENTO INDIRIZZO CONTENUTO IN DEP1 DI '03'
<
INDIR. CONTENUTO IN DEP1 ≷ INDIR. ULTIMO ELEMENTO
≥
STAMPA DEGLI ELEMENTI ORDINATI
STOP

Programma

Poichè risulta difficoltoso stabilire a priori il numero delle costanti, dei depositi, e delle aree di lavoro necessarie per la realizzazione del programma, per esse viene riservata una zona di memoria di indirizzo 0300 (768 in decimale), e di lunghezza indefinita.
Le istruzioni del linguaggio macchina costituenti il programma vengono registrate in memoria a partire dall'indirizzo 0040, e da tale indirizzo devono pure essere eseguite.

Il 'foglio di programmazione' contenente il programma è illustrato nelle figure 4.16 e 4.17, mentre la mappa della zona di memoria di indirizzo 0300 contenente i valori delle costanti ed i valori iniziali dei depositi e delle aree di lavoro è riportata nelle figure 4.14 e 4.15.

SALTI	INDIRIZZO ISTRUZIONE (1-4)	CODICE SIMBOL. (5-7)	COD. OPER. (8-9)	LUNG. (10-11)	1° OPERANDO (12-15)	2° OPERANDO (16-19)	OSSERVAZIONI
	0300		03	62			COSTANTE INDN - INDIRIZZO ULTIMO ELEMENTO
	0302		03	29			COSTANTE IND1 - INDIRIZZO PRIMO ELEMENTO
	0304		03				COSTANTE ESAD.'03' - LUNGHEZZA ELEMENTI
	0305		70	70	7070	7070	
	030B		75	83	7584	7585	COSTANTE 'TESTATA' - CONTIENE LA SCRITTA ALFA
	0311		93	79	7086	8974	NUMERICA "ELEMENTI ORDINATI"
	0317		79	85	7193	7970	
	031D		04				COSTANTE ESAD'04'- LUNGHEZZA ELEMENTO RIGA DI STAMPA
	031E		03	6A			INDIRIZZO 1° ELEMENTO RIGA DI STAMPA
	0320		03	7A			INDIRIZZO ULTIMO ELEMENTO RIGA DI STAMPA
	0322		00	00			DEPOSITO DEP1
	0324		00	00			DEPOSITO DEP2
	0326		60	60	60		AREA DI SCAMBIO
	0329						
	032F						
	0335						DATI DA ORDINARE
	033B						
	0341						

Fig. 4.14 - Esempio 4 - Costanti e Depositi

SALTI	INDIRIZZO ISTRUZIONE (1 2 3 4)	CODICE SIMBOL. (5 6 7)	COD. OPER. (8 9)	LUNG. (10 11)	1° OPERANDO (12 13 14 15)	2° OPERANDO (16 17 18 19)	OSSERVAZIONI
	0347						
	034D						
	0353						DATI DA ORDINARE
	0359						
	035F						
	0365						
	036B						
	0371						RIGA DI STAMPA (24 BYTES)
	0377						
	037D						

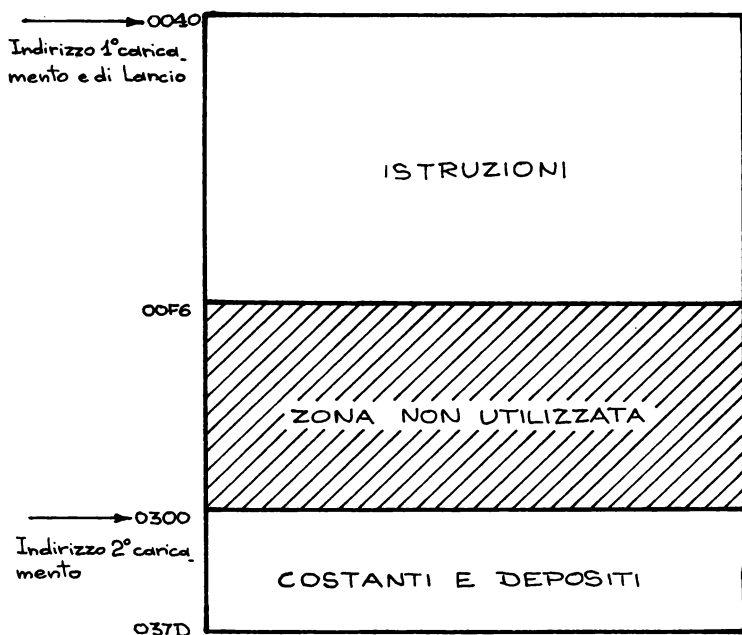
Fig. 4.15 - Esempio 4 - Costanti e Depositi (cont.)

SALTI	INDIRIZZO ISTRUZIONE (1-4)	CODICE SIMBOL. (5-7)	ISTRUZIONE: COD. OPER. (8-9)	LUNG. (10-11)	1° OPERANDO (12-15)	2° OPERANDO (16-19)	OSSERVAZIONI
	0040	IAL	B2	3B	0329		INTRODUZIONE DI 60 CARATTERI ALFANUMERICI (20 NUMERI)
	0044	TRA	A5	01	0322	0302	TRASFERIMENTO INDIR. 1° ELEMENTO NEL DEPOSITO DEP1
	004A	TRA	A5	01	0324	0322	TRASFERIMENTO VALORE DI DEP1 NEL DEPOSITO DEP2
	0050	AES	A3	31	0324	0304	SOMMA DI '03' AL CONTENUTO DEL DEPOSITO DEP2
	0056	TRA	A5	03	005E	0322	TRASFER. DEL CONTENUTO DI DEP1 E DEP2 NELL'ISTRUZ. SUCC.
	005C	CAL	A7	22	0000	0000	CONFRONTO ELEM. DI INDIRIZZI CONTENUTI NEI DEP. DEP1 E DEP2
	0062	SAL	B5	D0	008A		SE RISULTATO DEL CONFRONTO É ≤ SALTO ALL'ISTRUZ. 008A
	0066	TRA	A5	01	0070	0322	TRASFERIM. CONTENUTO DI DEP1 NELL'ISTRUZIONE SUCCESSIVA
	006C	TRA	A5	02	0326	0000	TRASFER. ELEMENTO DI INDIR. CONT. IN DEP1 NEL DEP. SCAMBIO
	0072	TRA	A5	03	007A	0322	TRASFER. CONTENUTO DI DEP1 E DEP2 NELL'ISTRUZ. SUCCESSIVA
	0078	TRA	A5	02	0000	0000	TRASF. ELEMENTO DI INDIR. DEP2 NELL'ELEMENTO DI INDIR. DEP1
	007E	TRA	A5	01	0086	0324	TRASFER. CONTENUTO DI DEP2 NELL'ISTRUZIONE SEGUENTE
	0084	TRA	A5	02	0000	0326	TRASFER. CONTENUTO DI SCAMBIO NELL'ELEM. DI INDIR. DEP2
	008A	AES	A3	31	0324	0304	SOMMO AL CONTENUTO DI DEP2 LA COSTANTE '03'
	0090	CLO	A6	01	0324	0300	CONFRONTO CONTENUTO DI DEP2 CON INDIR. ULTIMO ELEM. INDN
	0096	SAL	B5	D0	0056		SE RISULTATO CONFRONTO ≤ SALTO ALL'ISTRUZIONE 0056
	009A	AES	A3	31	0322	0304	SOMMO AL CONTENUTO DI DEP1 LA COSTANTE '03'
	00A0	CLO	A6	01	0322	0300	CONFRONTO CONTENUTO DI DEP1 CON INDIR. ULT. EL. INDN

Fig. 4.16 - Esempio 4 - Programma

SALTI	INDIRIZZO ISTRUZIONE (1-4)	CODICE SIMBOL. (5-7)	COD. OPER. (8-9)	LUNG. (10-11)	1° OPERANDO (12-15)	2° OPERANDO (16-19)	OSSERVAZIONI
	00A6	SAL	B5	40	004A		SE RISULTATO CONFRONTO É < SALTO ALL'ISTRUZIONE 004A
	00AA	TRA	A5	17	0365	0305	TRASFERIM. 'TESTATA' NELLA RIGA DI STAMPA
	00B0	OAL	B4	17	0365		STAMPA DELLA 'TESTATA'
	00B4	TRA	A5	16	0366	0365	SCRITTURA DI SPAZI (b̸) IN TUTTA LA RIGA DI STAMPA
	00BA	OAL	B4	17	0365		STAMPA DELLA RIGA CONTENENTE SOLTANTO SPAZI
	00BE	TRA	A5	02	036A	0329	SCRITTURA NELLA RIGA DELL'ELEMENTO DI INDIR. 2° OPERANDO
	00C4	AES	A3	31	00C2	0304	SOMMO '03' ALL'INDIRIZZO 2° OPERANDO ISTRUZIONE PRECED.
	00CA	AES	A3	31	00C0	031D	SOMMO '04' ALL'INDIRIZZO 1° OPERANDO ISTRUZ. PREC. DI IND. 00BE
	00D0	CLO	A6	01	00C0	0320	CONFRONTO INDIR. 1° OPER. ISTRUZ. 00BE CON IND. FINE RIGA STAMPA
	00D6	SAL	B5	D0	00BE		SE RISULTATO CONFRONTO ≤ SALTO ALL'ISTRUZIONE 00BE
	00DA	OAL	B4	17	0365		STAMPA DELLA 'RIGA' CONTENENTE 5 ELEM. ORDINATI
	00DE	CLO	A6	01	00C2	0362	CONFRONTO INDIR. 2° OPER. ISTR. 00BE CON IND. ULT. EL. IND N
	00E4	SAL	B5	80	00F2		SE RISULTATO CONFRONTO É = SALTO ALL'ISTRUZIONE 00F2
	00E8	TRA	A5	01	00C0	031E	TRASFER. INDIRIZZO 1° ELEMENTO DELLA 'RIGA DI STAMPA'
	00EE	SAL	B5	F0	00BE		SALTO INCONDIZIONATO ALL'ISTRUZIONE 00BE
	00F2	ALT	B7	00	0004		FINE DEL PROGRAMMA

Fig. 4.17 - Esempio 4 - Programma (cont.)

La struttura adottata in tale programma è del seguente tipo:

Il programma è lungo 307 caratteri (182 per le istruzioni e 125 per le costanti) però la occupazione della memoria è globalmente di 829 bytes in quanto la 'zona non utilizzata', tra le istruzioni ed i dati costanti, i depositi, e le aree è di 522 bytes.

E' da notare il particolare uso della istruzione di trasferimento di indirizzo 00B4 fatto nel programma. Lo scopo è quello di 'pulire' l'area di stampa, e cioè di scrivere in tutte le posizioni di tale area il carattere b̸.

Questa istruzione di trasferimento di una zona su se stessa scalata di una posizione realizza esattamente tale scopo; infatti, poichè il trasferimento inizia dalla sinistra, il primo carattere (che è b̸) viene registrato sul secondo, il secondo (che ora è anch'esso b̸) viene registrato sul terzo, e così via per la lunghezza indicata nella istruzione.

Si immagini di eseguire il programma introducendo i seguenti nu-meri:

85+, 88+, 12-, 33-, 25+, 36+, 11+, 95-, 54-, 51+,

18-, 68+, 19-, 88-, 43+, 35+, 12+, 10+, 99+, 58+

Alla fine della sua esecuzione si sarà ottenuta la seguente stampa:

```
85+88+12-33-25+36+11+95-54-51+18-68+19-88-43+35+12+10+99+58+
      ELEMENTI ORDINATI

     95- 88- 54- 33- 19-
     18- 12- 10+ 11+ 12+
     25+ 35+ 36+ 43+ 51+
     58+ 68+ 85+ 88+ 99+

                            ALT 000004
```

CAPITOLO 5

SOTTOPROGRAMMI

5.1 GENERALITA'

5.2 IL RIENTRO DA UN SOTTOPROGRAMMA

5.3 I PARAMETRI DI UN SOTTOPROGRAMMA

5.4 UN ESEMPIO DI SOTTOPROGRAMMA

5.1 GENERALITA'

Molto spesso un programmatore deve realizzare un programma che richiede in parecchi suoi punti la esecuzione di uno stesso insieme di operazioni da effettuarsi ogni volta su dati diversi (ad esempio la estrazione della radice quadrata, il calcolo del logaritmo, la ricerca della componente di massimo valore di un dato vettore, ecc.).

La soluzione che consiste nel ripetere ogni volta nel programma la sequenza di istruzioni che realizza le operazioni richieste è senza dubbio la più ovvia, però è anche la più dispendiosa sia come tempo di codifica che come occupazione di memoria.

Appare subito evidente che la soluzione più funzionale è quella di scrivere una sola volta la sequenza delle istruzioni richieste e di utilizzarla tutte le volte che è necessario richiamandola in modo opportuno.

Una sequenza di istruzioni che svolge una funzione e che nel corso dell'esecuzione di un programma, detto 'programma-principale' (main-program), viene utilizzato più volte in punti diversi viene detta "sottoprogramma".

L'insieme delle istruzioni che costituisce il 'sottoprogramma' deve essere scritto una sola volta, ad esempio in coda al 'programma principale', e la sua esecuzione viene da questo avviata mediante una operazione comunemente detta di 'richiamo' del sottoprogramma.

Il 'richiamo' di un sottoprogramma consiste in un salto incondizionato all'indirizzo della prima istruzione del sottoprogramma e nell'eventuale passaggio ad esso di uno o più dati, detti 'parametri'.

Il passaggio dei parametri può essere realizzato in vari modi, ad esempio depositando i loro valori o in zone di memoria prefissate e quindi note al sottoprogramma oppure fornendo al sottoprogramma i loro indirizzi.

Mentre il richiamo di un sottoprogramma non presenta alcuna dif ficoltà in quanto l'indirizzo della sua prima istruzione è fisso qualsiasi sia il punto di richiamo, il meccanismo del 'rientro' dal sottoprogramma al programma principale risulta meno semplice, in quanto tale 'rientro' deve avvenire ad indirizzi di versi in funzione dell'indirizzo del relativo 'richiamo'.

Nella figura 5.1 è schematizzato un programma nel quale viene richiesta in 3 punti distinti l'esecuzione delle stesse sequenze di operazioni (ad esempio il calcolo della radice quadrata).

Nella figura 5.2 è schematizzato lo stesso programma nel quale però è stato realizzato un 'sottoprogramma' per effettuare il calcolo della radice quadrata.

Tale sottoprogramma viene 'richiamato' dal programma principale 3 volte da 3 punti diversi, e quindi esistono 3 'rientri' al programma principale in 3 punti diversi.

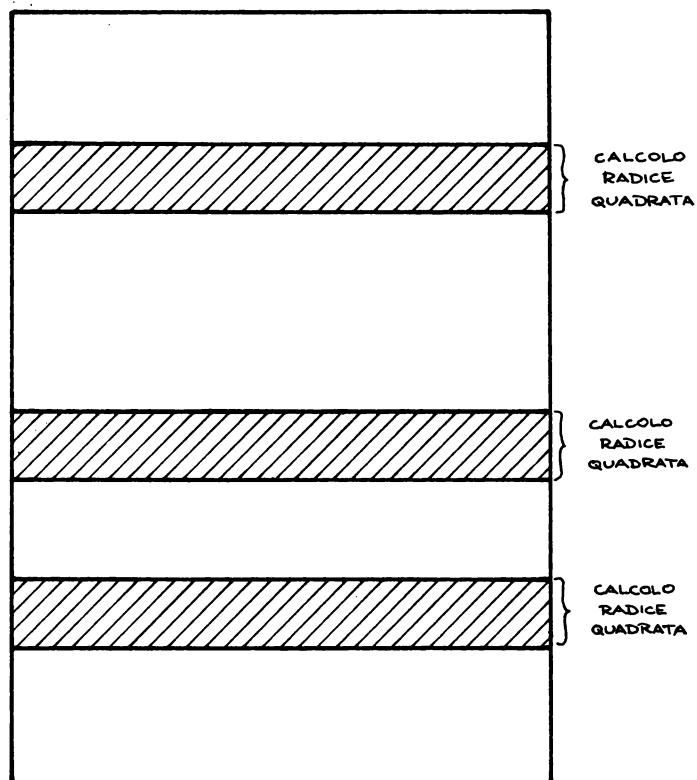



Fig. 5.1 - Mappa di un programma che richiede il calcolo della radice quadrata in 3 punti distinti

Nei paragrafi seguenti vengono risolte le principali difficoltà che sorgono dall'uso dei sottoprogrammi, e cioè:

a - determinazione dell'indirizzo del programma principale al quale deve avvenire il 'rientro' dal sottoprogramma

b - determinazione delle posizioni di memoria nelle quali si trovano i 'parametri' ('parametri in ingresso') passati dal programma principale al sottoprogramma

c - determinazione delle posizioni di memoria nelle quali devono essere forniti i 'risultati' ('parametri in uscita') passati dal sottoprogramma al programma principale.

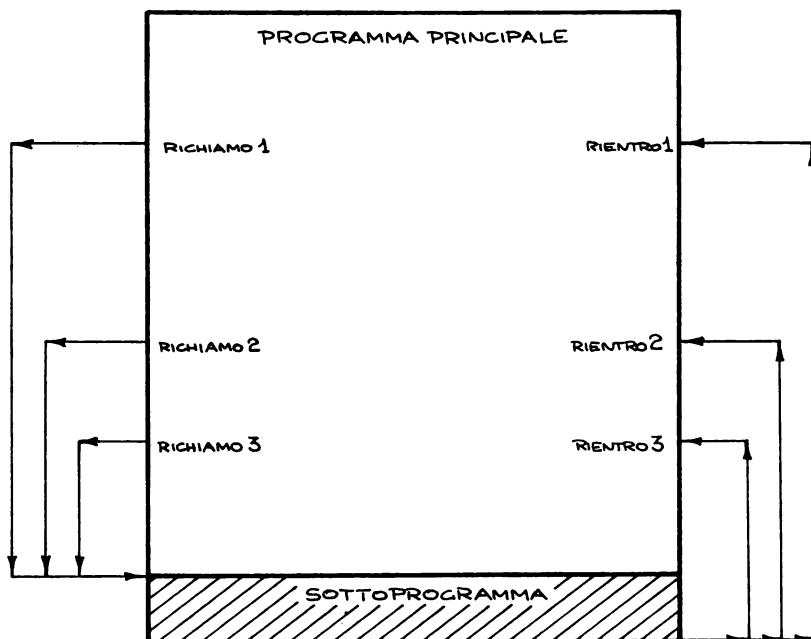



Fig. 5.2 - Mappa di un programma con 'sottoprogramma' per il calcolo della radice quadrata

5.2 IL 'RIENTRO' DA UN SOTTOPROGRAMMA

Il meccanismo del 'rientro', nella versione base del calcolatore CND, viene realizzato automaticamente nel seguente modo:

- al momento del 'richiamo' del sottoprogramma, eseguito mediante una istruzione 'SME' (salto con memorizzazione) viene automaticamente memorizzato in un deposito della 'zona riservata della memoria' (bytes 00-01) l'indirizzo della istruzione del programma principale successiva alla SME stessa;

- questo indirizzo, al termine del sottoprogramma, deve essere trasferito a cura del programmatore nella istruzione di 'salto incondizionato ' (ultima istruzione di ogni sottoprogramma) realizzando così il rientro all'istruzione del programma principale successiva a quella che ha effettuato il richiamo.

Nella figura 5.3 è riportato un 'programma principale' che richiama due volte un 'sottoprogramma'.

L'indirizzo di inizio del sottoprogramma è 0B00 (2816 in decimale) e l'indirizzo di rientro che compare nella sua ultima istruzione (istruzione SAL di indirizzo 0B90) viene scritto automaticamente ad ogni esecuzione a cura della precedente istruzione di trasferimento TRA, che lo preleva ogni volta dai primi due bytes della memoria (deposito di indirizzo 0000).

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	0	2	1	A	S	M	E	B	6	0	0	0	B	0	0					PRIMO RICHIAMO DEL 'SOTTOPROGRAMMA'
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	0	9	1	E	S	M	E	B	6	0	0	0	B	0	0					SECONDO RICHIAMO DEL 'SOTTOPROGRAMMA'
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
RICHIAMO 1 / RICHIAMO 2 / RIENTRO 1 / RIENTRO 2	0	A	F	C	A	L	T	B	7	0	0	0	A	A	A					FINE DEL 'PROGRAMMA PRINCIPALE'
	0	B	0	0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	PRIMA ISTRUZIONE DEL 'SOTTOPROGRAMMA'
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	0	B	8	A	T	R	A	A	5	0	1	0	B	9	2	0	0	0	0	TRAFER. INDIRIZZO DI RIENTRO NELLA ISTRUZIONE SUCCESS.
	0	B	9	0	S	A	L	B	5	F	0									ULTIMA ISTRUZIONE DEL 'SOTTOPROGRAMMA'

Fig. 5.3 - Richiamo e Rientro di un SOTTOPROGRAMMA

5.3 I PARAMETRI DI UN SOTTOPROGRAMMA

Generalmente è necessario fornire a ciascun sottoprogramma alcuni dati (detti 'parametri') sui quali esso effettua le operazioni richieste (ad esempio un sottoprogramma per il calcolo della radice quadrata ha come parametro di ingresso il radicando, mentre un sottoprogramma per la risoluzione di una equazione di secondo grado ha come parametri di ingresso i tre coefficienti).

Si è detto 'generalmente' perchè per la esecuzione di particolari sottoprogrammi, quali ad esempio quelli che provocano unicamente la stampa di testate, di messaggi di avviso, ecc., non è necessaria la conoscenza di alcun dato.

Il numero dei parametri da passare ad ogni sottoprogramma non è costante, ma dipende soltanto dalle funzioni svolte dal particolare sottoprogramma considerato.

Nella versione base del calcolatore CND il problema del passaggio dei parametri ad un sottoprogramma può essere risolto scrivendo la 'lista dei parametri' in una posizione convenzionale del programma principale, ad esempio immediatamente dopo l'istruzione SME che effettua il richiamo del sottoprogramma.

Il sottoprogramma è così in grado di risalire ai valori dei parametri, in quanto esso conosce l'indirizzo di inizio della 'lista' (che è l'indirizzo memorizzato nei primi 2 bytes della 'zona riservata della memoria' a cura dell'istruzione SME).

Con l'espressione 'lista dei parametri' si intende una successione di bytes contenenti i valori dei parametri oppure i loro indirizzi di memoria.

Normalmente tale lista contiene gli indirizzi di memoria dei parametri ciascuno dei quali occupa pertanto 2 bytes.

Un discorso analogo vale anche per i parametri che vengono passati dal sottoprogramma al programma principale, e cioè per i risultati del sottoprogramma (detti anche 'parametri in uscita').

Gli indirizzi di questi nuovi parametri saranno normalmente gli ultimi della 'lista', e seguiranno quelli dei 'parametri in ingresso'.

Poichè la lista dei parametri non deve essere considerata durante la esecuzione del programma principale, il sottoprogramma dovrà effettuare il rientro all'indirizzo memorizzato dell'istruzione di richiamo SME aumentato del numero dei bytes occupati dalla lista stessa.

Perciò, alla fine di un sottoprogramma che richiede dei parametri, oltre all'istruzione di trasferimento TRA deve comparire anche una istruzione di addizione esadecimale AES, che serve per compilare nell'ultima istruzione SAL (salto incondizionato) il valore esatto dell'indirizzo di rientro.

Per chiarire tale procedimento viene descritto nel paragrafo seguente un esempio di programma con richiami di un sottoprogramma.

Occorre tenere presente che, essendo l'unico scopo dell'esempio quello di chiarire l'uso dei sottoprogrammi, è stato scelto un problema particolarmente semplice che non presenta alcuna difficoltà di calcolo.

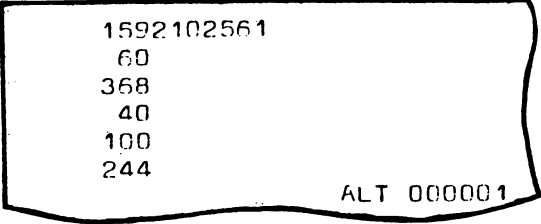
5.4 UN ESEMPIO DI SOTTOPROGRAMMA

Il problema da risolvere è il seguente:

costruire un programma che quadruplichi i valori di 5 numeri decimali.

Appare subito evidente che per effettuare i calcoli necessari a quadruplicare i numeri dati è conveniente costruire un unico sottoprogramma, che può essere richiamato tante volte quanti sono i numeri da elaborare.

Questo sottoprogramma ha un unico 'parametro in ingresso', che è il valore del numero da quadruplicare, ed ha anche un unico 'parametro in uscita', che è il valore quadruplicato.

Il programma principale che realizza il problema dato è riportata nella Figura 5.4, mentre il sottoprogramma è riportato nella Figura 5.6.

I depositi necessari al programma principale (25 bytes) si trovano nella memoria a partire dall'indirizzo 0100, e la loro mappa è riportata nella Figura 5.5..

Il sottoprogramma è memorizzato a partire dall'indirizzo OFOO.

L'esecuzione completa di tale programma utilizzando come dati i valori:

15 92 10 25 61

darebbe luogo alla seguente stampa:

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	0	0	4	0	I	A	L	B	2	0	9	0	1	0	0					INTRODUZIONE DEI 5 NUMERI DATI DI 2 CIFRE
	0	0	4	4	S	M	E	B	6	0	0	0	F	0	0					1° RICHIAMO DEL SOTTOPROGRAMMA
	0	0	4	8				0	1	0	0	0	1	0	A					LISTA DEI PARAMETRI (INDIRIZZO DATO, INDIRIZZO RISULTATO)
	0	0	4	C	0	A	L	B	4	0	2	0	1	0	A					STAMPA DEL VALORE QUADRUPLICATO OTTENUTO
	0	0	5	0	S	M	E	B	6	0	0	0	F	0	0					2° RICHIAMO DEL SOTTOPROGRAMMA
	0	0	5	4				0	1	0	2	0	1	0	D					LISTA DEI PARAMETRI (INDIRIZZO DATO, INDIRIZZO RISULTATO)
	0	0	5	8	0	A	L	B	4	0	2	0	1	0	D					STAMPA DEL VALORE QUADRUPLICATO OTTENUTO
	0	0	5	C	S	M	E	B	6	0	0	0	F	0	0					3° RICHIAMO DEL SOTTOPROGRAMMA
	0	0	6	0				0	1	0	4	0	1	1	0					LISTA DEI PARAMETRI (INDIRIZZO DATO, INDIRIZZO RISULTATO)
	0	0	6	4	0	A	L	B	4	0	2	0	1	1	0					STAMPA DEL VALORE QUADRUPLICATO OTTENUTO
	0	0	6	8	S	M	E	B	6	0	0	0	F	0	0					4° RICHIAMO DEL SOTTOPROGRAMMA
	0	0	6	C				0	1	0	6	0	1	1	3					LISTA DEI PARAMETRI (INDIRIZZO DATO, INDIRIZZO RISULTATO)
	0	0	7	0	0	A	L	B	4	0	2	0	1	1	3					STAMPA DEL VALORE QUADRUPLICATO OTTENUTO
	0	0	7	4	S	M	E	B	6	0	0	0	F	0	0					5° RICHIAMO DEL SOTTOPROGRAMMA
	0	0	7	8				0	1	0	8	0	1	1	6					LISTA DEI PARAMETRI (INDIRIZZO DATO, INDIRIZZO RISULT.)
	0	0	7	C	0	A	L	B	4	0	2	0	1	1	6					STAMPA DEL VALORE QUADRUPLICATO OTTENUTO
	0	0	8	0	A	L	T	B	7	0	0	0	0	0	1					FINE DEL PROGRAMMA PRINCIPALE

Fig. 5.4 - Programma Principale

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	0	1	0	0																DEPOSITI PER I 5 NUMERI DATI DI 2 CIFRE (10 BYTES)
	0	1	0	6																
	0	1	0	A																DEPOSITI PER I 5 VALORI QUADRUPLICATI
	0	1	1	0																(15 BYTES)
	0	1	1	6																

Fig. 5.5 - Depositi

SALTI	INDIRIZZO ISTRUZIONE (1-4)	CODICE SIMBOL. (5-7)	COD. OPER. (8-9)	LUNG. (10-11)	1° OPERANDO (12-15)	2° OPERANDO (16-19)	OSSERVAZIONI
	0F00	SES	A4	55	0F40	0F40	AZZERAMENTO DEPOSITO PER VALORE QUADRUPLICATO
	0F06	TRA	A5	01	0F10	0000	TRASFER. INDIRIZZO DEL DATO NELL'ISTRUZIONE SUCCESSIVA
	0F0C	TRA	A5	01	0F40	[ ]	PRELEVAMENTO DEL PARAMETRO IN INGRESSO
	0F12	ADE	A1	22	0F40	0F40	SOMMA DEL DATO A SE STESSO (RADDOPPIO DEL VALORE)
	0F18	ADE	A1	22	0F40	0F40	SOMMA DEL NUOVO VALORE A SE STESSO (QUADRUPLIC. DEL DATO)
	0F1E	AES	A3	31	0F10	0F43	CALCOLO INDIRIZZO RISULTATO
	0F24	TRA	A5	01	0F2C	0F10	TRASFER. INDIRIZZO RISULTATO NELL'ISTRUZIONE SUCCESSIVA
	0F2A	TRA	A5	02	[ ]	0F40	SCRITTURA DEL PARAMETRO IN USCITA
	0F30	AES	A3	31	0F10	0F43	CALCOLO DELL'INDIRIZZO DI RIENTRO
	0F36	TRA	A5	01	0F3E	0F10	TRASFER. INDIRIZZO DI RIENTRO NELL'ISTRUZ. SUCCESSIVA
	0F3C	SAL	B5	F0	[ ]		SALTO INCONDIZIONATO ALL'INDIRIZZO DI RIENTRO
	0F40		X	X	X		DEPOSITO PER IL RISULTATO (VALORE QUADRUPLICATO)
	0F43		04				COSTANTE ESADECIMALE

Fig. 5.6 - Sottoprogramma

CAPITOLO 6

ESTENSIONE DEL CALCOLATORE CND

6.1 I REGISTRI INDICE

6.2 L'INDIRIZZAMENTO INDIRETTO DELLA MEMORIA

6.3 LE ISTRUZIONI PER I REGISTRI INDICE

6.4 USO DEI REGISTRI INDICE NEI SOTTOPROGRAMMI

6.5 RICOLLOCABILITA' DEI PROGRAMMI

6.1 I REGISTRI INDICE

In questo capitolo viene descritta una versione più estesa del calcolatore CND.
Questa nuova versione, che è un ampliamento della versione base descritta nei capitolo precedenti, prevede oltre ad una maggiore capacità della memoria a nuclei magnetici l'uso di particolari dispositivi detti 'registri indice'.

Il calcolatore dispone di 16 registri, aventi la lunghezza di 2 bytes (16 bits) ciascuno, identificati mediante i numeri interi da 0 a 15 (in esadecimale da 0 a F).

Fisicamente tali 'registri indice' sono dei dispositivi elettronici realizzati con nuclei magnetici e sono allocati nella Zona Riservata della Memoria (figura 1.9) rispettivamente nelle posizioni 00 ÷ 1F (32 bytes, 16 registri).

I registri indice, oltre a fornire un nuovo sistema di indirizzamento della memoria, permettono di ottenere la ricollocabilità dei programmi, facilitano il collegamento tra programma principale e diverse classi di sottoprogrammi e rendono possibile un trattamento più funzionale delle tabelle di dati.

Le informazioni numeriche contenute nei registri indice vengono sempre rappresentate in esadecimale (o binario).

Ad esempio il numero 332 è contenuto in un registro indice nel seguente modo:

Nella figura 6.1 è riportata una mappa aggiornata della Zona Riservata della Memoria contenente i 16 'registri indice'.

REGISTRO 0	REGISTRO 1	REGISTRO 2	REGISTRO 3
REGISTRO 4	REGISTRO 5	REGISTRO 6	REGISTRO 7
REGISTRO 8	REGISTRO 9	REGISTRO A	REGISTRO B
REGISTRO C	REGISTRO D	REGISTRO E	REGISTRO F
REGISTRO STATO DEL PROGRAMMA		IND. CARICAMEN.	IND. LANCIO
A DISPOSIZIONE DELLA 'ROM'			
			3F

Fig. 6.1 - Mappa della 'Zona Riservata della Memoria'

Le istruzioni che operano sui 'registri indice' sono descritte nel paragrafo 6.3.

6.2 L'INDIRIZZAMENTO INDIRETTO DELLA MEMORIA

Nella versione base del calcolatore CND l'indirizzamento viene effettuato in modo diretto; cioè a ciascuno dei 4096 bytes costituenti la memoria è associato l'indirizzo effettivo, che compare nelle istruzioni, consistente in un numero esadecimale compreso tra 0 e FFF (cioè tra 0 ed il valore decimale 4095).

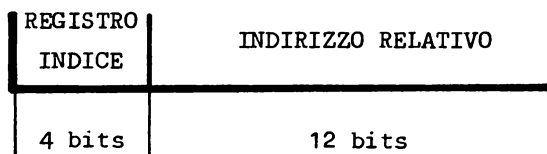
Poichè in tutte le istruzioni del linguaggio macchina il campo riservato al valore dell'indirizzo è lungo 2 bytes (16 bits), l'indirizzamento di tipo diretto è possibile sino a che le dimensioni della memoria del calcolatore non superano 65.536 bytes (cioè 2^{16}).

Se la memoria del calcolatore ha un numero di posizioni superiore a 65.536, per poter ancora indirizzare direttamente tutta la memoria sarebbe necessario un numero di bits superiore a 16, e ciò comporterebbe un aumento della lunghezza del campo riservato all'indirizzo e quindi della lunghezza delle istruzioni.

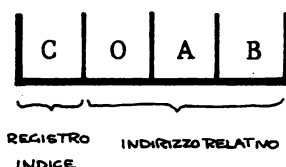
Inoltre un programma redatto in linguaggio macchina con indirizzamento di tipo diretto deve obbligatoriamente essere sempre registrato nelle stesse posizioni di memoria per poter essere eseguito in modo corretto.

Nel metodo di indirizzamento indiretto che adotteremo nel seguito, il valore effettivo dell'indirizzo di memoria si ottiene sommando il contenuto di un 'registro indice' al valore di indirizzo relativo specificato dal programmatore.nell'istruzione.

Il campo riservato ai valori degli indirizzi è sempre lungo 2 bytes ed ha la seguente struttura:

REGISTRO INDICE	INDIRIZZO RELATIVO
4 bits	12 bits

Dei 16 bits (2 bytes) del campo riservato ad un indirizzo, i primi 4 servono quindi per specificare un 'registro indice' (da 0 a F), i restanti 12 bits servono per fornire un valore di 'indirizzo relativo' ($\leq$ 4095 , $\leq$ FFF).

Il calcolo dell'indirizzo effettivo, e cioè la somma del contenuto del registro indice e del valore dell'indirizzo relativo, viene effettuato automaticamente dal calcolatore al momento dell'esecuzione della istruzione considerata.

Si consideri ad esempio il seguente indirizzo:

Se nel registro C è contenuto il valore 16.384, l'indirizzo effettivo si ottiene come risultato della seguente somma:

16384 +	contenuto registro indice
171	indirizzo relativo
16555	indirizzo effettivo

Con l'indirizzamento indiretto è possibile indirizzare qualsiasi valore di memoria mantenendo costante la lunghezza del campo riservato agli indirizzi (se è necessario basterà ampliare la capacità dei 'registri indice'), e possono essere facilmente eliminati anche altri inconvenienti riscontrati nell'indirizzamento diretto.

Osserviamo che la lunghezza dei registri indice del calcolatore CND, è limitata a 2 bytes; pertanto l'indirizzo massimo esprimibile è sostanzialmente quello che si potrebbe esprimere con l'indirizzamento diretto. Per permettere l'indirizzamento di una memoria più ampia sarebbe necessario aumentare la lunghezza dei registri indice a 3 o 4 bytes.

E' anche opportuno precisare che con l'introduzione dei registri indice il caricamento ed il lancio di un programma avvengono in modo lievemente diverso rispetto a quello descritto nel paragrafo 1.7.

La differenza sta nel fatto che, con questo nuovo metodo di indirizzamento, l'operatore deve battere sulla tastiera esadecimale il contenuto del registro indice utilizzato nel programma per l'indirizzamento.

Come conseguenza di questa modifica è necessario che il calcolatore conosca il nome del registro indice utilizzato nel programma ed in cui caricare quindi il valore introdotto dal programmatore.

Quindi i punti c) dei due programmi di caricamento e di lancio descritti nei paragrafi 1.7.1 e 1.7.2 diventano rispettivamente:

c) Battitura sulla tastiera esadecimale di 5 cifre esadecimali, delle quali la 1° rappresenta il nome del registro indice utilizzato per l'indirizzamento e le 4 rimanenti rappresentano il valore che viene in esso caricato.

c) Battitura sulla tastiera esadecimale di 4 cifre esadecimali che verranno automaticamente sommate al contenuto del registro indice specificato nel programma di caricamento, ottenendo così il valore esatto dell'indirizzo di lancio.

6.3 LE ISTRUZIONI PER I REGISTRI INDICE

In questo paragrafo vengono descritte le istruzioni disponibili per operare con i 'registri indice'.

Tali istruzioni svolgono le seguenti funzioni fondamentali:

- trasferimento del contenuto di una zona di memoria in un registro
- trasferimento del contenuto di un registro in una zona di memoria
- somma esadecimale del contenuto di una zona di memoria al contenuto di un registro
- sottrazione esadecimale del contenuto di una zona di memoria dal contenuto di un registro
- confronto logico del contenuto di un registro con il contenuto esadecimale di una zona di memoria

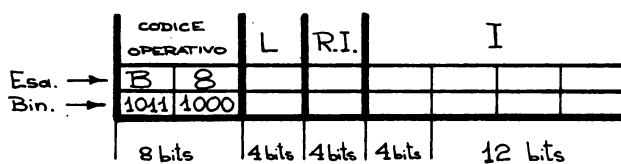
La loro lunghezza è di 4 bytes; le loro funzioni ed i loro formati vengono ora descritti in dettaglio:

ISTRUZIONI	CODICE SIMBOLICO	CODICE OPERATIVO
Trasferimento memoria-registro	MRE	B8
Trasferimento registro-memoria	RME	B9
Addizione registro-memoria	ARM	BA
Sottrazione registro-memoria	SRM	BB
Confronto registro-memoria	CRM	BC

6.3.1 Trasferimento memoria-registro

Funzione

Questa istruzione è di formato B.
Essa provoca il trasferimento del contenuto di una zona di memoria in un registro indice.
Prima della scrittura del nuovo dato, il registro indice viene azzerato.

La lunghezza della zona di memoria da trasferire può essere di 1 o 2 bytes, pertanto i soli valori che possono comparire in L sono 0 e 1, in caso contrario l'istruzione non viene eseguita e la elaborazione si arresta.

Codice operativo = B8 Codice simbolico = MRE

L = lunghezza, in bytes, diminuita di 1 del dato che deve essere trasferito

R.I. = nome (da 0 a F) del registro indice nel quale viene trasferito il dato

I = indirizzo espresso in modo indiretto del dato da trasferire

Fig. 6.2 - Formato del 'Trasferimento memoria-registro'

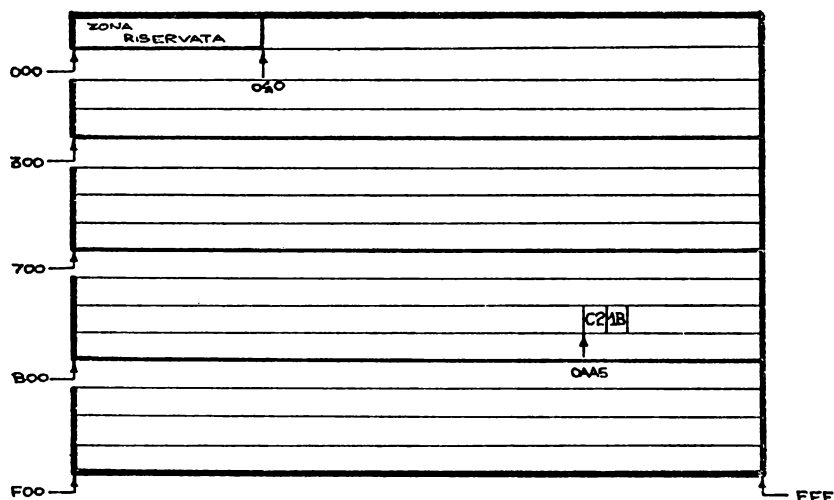
Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

Esempio

Si consideri la seguente istruzione:

Se il registro B contiene il valore esadecimale 0A00, essa provoca il trasferimento del contenuto della zona di memoria di indirizzo 0A00 + 0A5 (e cioè 0AA5) lunga 2 bytes nel registro A.

Se la mappa della memoria prima della esecuzione di tale istruzione è:

Il registro A, dopo la esecuzione della istruzione, conterrà il valore esadecimale:

C21B

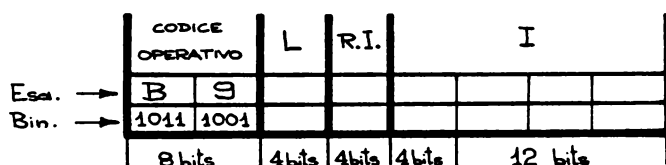
6.3.2 Trasferimento registro-memoria

Funzione

Questa istruzione è di formato B.
Essa provoca il trasferimento del contenuto di un registro in una zona di memoria.
Il contenuto del registro può essere trasferito interamente (2 bytes) oppure può essere trasferita soltanto la metà meno significativa (1° byte di destra).

La lunghezza L esprime il numero di bytes del registro che vengono trasferiti, pertanto essa può assumere soltanto i valori 0 ed 1, in caso contrario l'istruzione non viene eseguita e la elaborazione si arresta.

Il contenuto del registro R.I. rimane inalterato dopo l'esecuzione di tale istruzione.

	CODICE OPERATIVO		L	R.I.	I			
Esa. →	B	9						
Bin. →	1011	1001						
	8 bits		4 bits	4 bits	4 bits	12 bits		

Codice operativo = B9 Codice simbolico = RME

L = lunghezza, in bytes, diminuita di 1 del contenuto del registro che deve essere trasferito

R.I. = nome (0÷F) del registro indice interessato nel trasferimento

I = indirizzo (espresso in modo indiretto) della zona di memoria nella quale viene trasferito il contenuto di R.I.

Fig. 6.3 - Formato del 'Trasferimento registro-memoria'

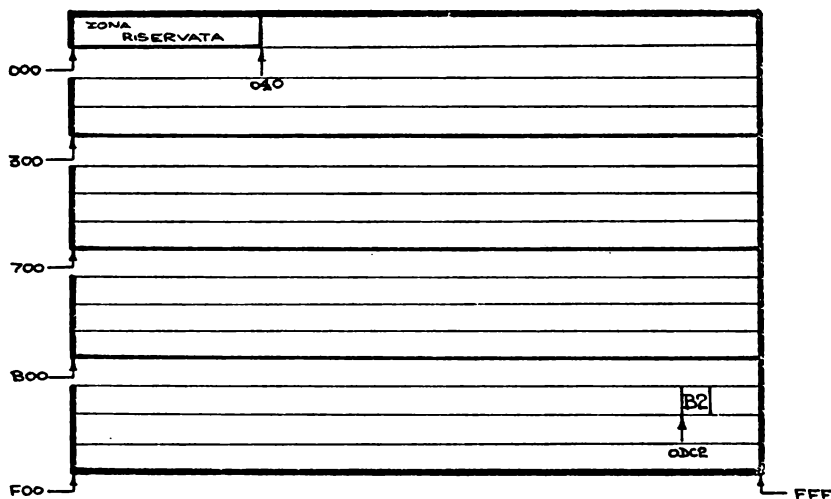
Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

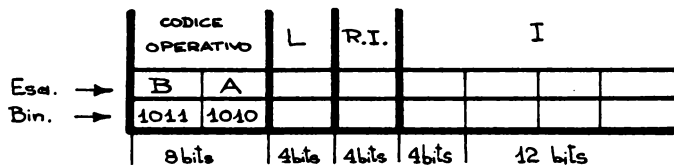
Esempio

Si consideri la seguente istruzione:

Se il registro C contiene il valore esadecimale A1B2 ed il registro B contiene il valore esadecimale 0D02, essa provoca il trasferimento del carattere di destra del registro C (B2) nella zona di memoria di indirizzo 0D02 + 00C0 = 0DC2.

Il contenuto del registro C dopo l'esecuzione della istruzione rimane inalterato, e la mappa della memoria è la seguente:

6.3.3 Addizione registro-memoria

Funzione

Questa istruzione è di formato B.
Essa provoca la somma esadecimale (o binaria) tra il contenuto di una zona di memoria ed il contenuto di un registro.
Il risultato della somma viene memorizzato nel registro, che perde così il dato contenuto in precedenza.
La lunghezza del dato da sommare al contenuto del registro può essere variabile, 1 o 2 bytes, e viene espressa nel campo L.
Pertanto i valori di L possono essere soltanto 0 o 1, in caso contrario l'istruzione non viene eseguita e la elaborazione si arresta.

Codice operativo = BA   Codice simbolico = ARM

L	= lunghezza, in bytes, diminuita di 1 del dato che viene sommato al registro R.I.
R.I.	= nome (da 0 a F) del registro indice al cui contenuto viene sommato il dato
I	= indirizzo (espresso in modo indiretto) del dato da sommare

Fig. 6.4 - Formato della 'Addizione memoria-registro'

Codice Condizione

I valori binari assunti dal Codice di Condizione hanno i seguenti significati:

00	se il risultato della somma è = 0
10	se il risultato della somma è > 0
11	se si è verificato il caso di OVERFLOW

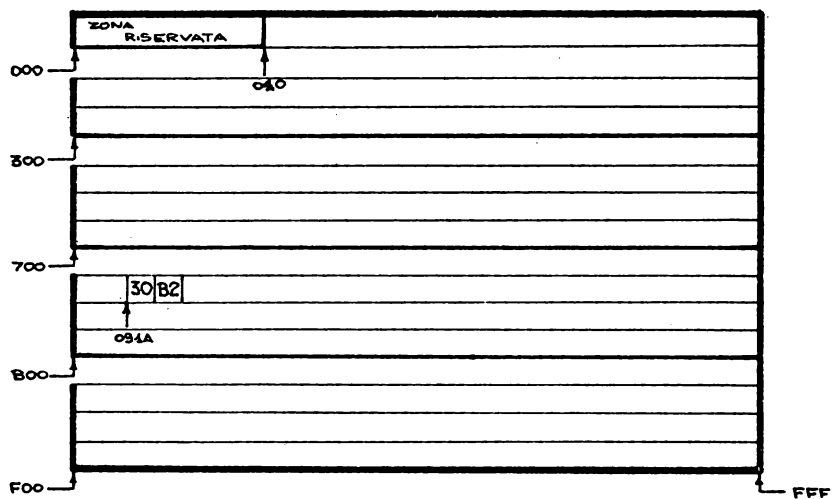
Il valore 01 non può essere assunto dal Codice di Condizione dopo una somma memoria-registro.

Esempio

Si consideri la seguente istruzione:

B A	1 9	8 3	1 A

Se prima della sua esecuzione il registro 9 contiene il valore esadecimale 1A08 ed il registro 8 contiene il valore esadecimale 0600, essa provoca la somma esadecimale del dato lungo 2 bytes memorizzato all'indirizzo 0600 + 031A = 091A al contenuto del registro 9.

Se la mappa della memoria prima della esecuzione di tale istruzione è:

Il contenuto del registro 9, dopo la sua esecuzione, sarà:

1A08 + 30B2 = 4ABA

mentre il contenuto della memoria rimane inalterato.

Il Codice Condizione assumerà il valore 10 in quanto il risultato della addizione è > 0.

6.3.4 Sottrazione registro-memoria

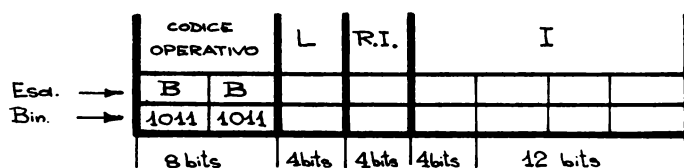
Funzione

Questa istruzione è di formato B.
Essa provoca la sottrazione esadecimale (o binaria) del contenuto di una zona di memoria dal contenuto di un registro.
Il risultato della sottrazione viene memorizzato nel registro, che perde così il dato contenuto in precedenza.

La lunghezza del dato da sottrarre al contenuto del registro può essere variabile, 1 o 2 bytes, e viene espresso nel campo L.

I valori di L possono essere quindi soltanto 0 o 1, in caso contrario l'istruzione non viene eseguita e la elaborazione si arresta.

Se il valore contenuto nel registro è maggiore od uguale al dato da sottrarre si ottiene la differenza fra i due, mentre se il valore contenuto nel registro è minore del dato da sottrarre si ottiene il complemento binario della differenza.
In quest'ultimo caso si ha la segnalazione di Overflow.

	CODICE OPERATIVO		L	R.I.	I			
Esa. →	B	B						
Bin. →	1011	1011						
	8 bits		4bits	4bits	4bits	12 bits		

Codice operativo = BB   Codice simbolico = SRM

L = lunghezza, in bytes, diminuita di 1 del dato che deve essere sottratto al registro R.I.

R.I. = nome (da 0 a F) del registro indice al cui contenuto viene sottratto il dato

I = indirizzo (espresso in modo indiretto) del dato da sottrarre

Fig. 6.5 - Formato della 'Sottrazione memoria-registro'

Codice Condizione

I valori binari assunti dal Codice di Condizione hanno i seguenti significati:

00	se il risultato della sottrazione è = 0
10	se il risultato della sottrazione è > 0
11	se si è verificato il caso di OVERFLOW

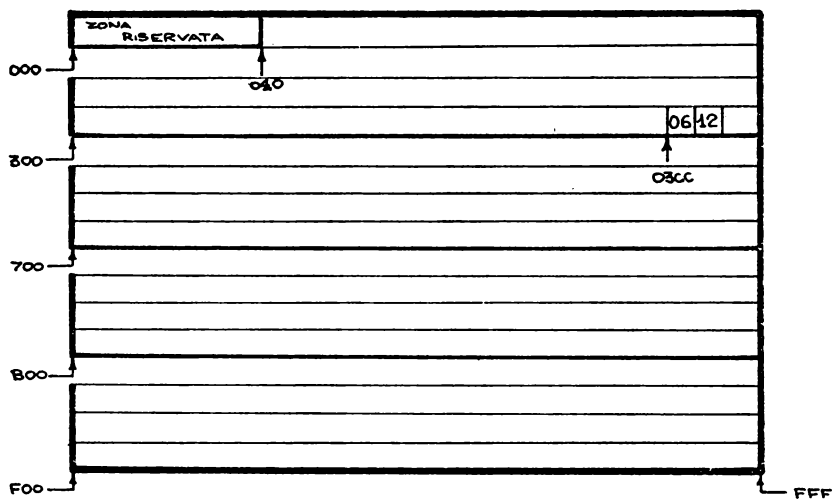
Il valore 01 non può essere assunto dal Codice di Condizione dopo una sottrazione memoria registro.

Esempio

Si consideri la seguente istruzione:

B	B	1	6	C	0	A	2

Se prima della sua esecuzione il registro 6 contiene il valore esadecimale B02A ed il registro C contiene il valore esadecimale 032A, essa provoca la sottrazione esadecimale del dato lungo 2 bytes memorizzato all'indirizzo 032A + 00A2 = 03CC al contenuto del registro 6.

Se la mappa della memoria al momento della esecuzione di tale istruzione è:

Il contenuto del registro 6, dopo la sua esecuzione, sarà:

B02A - 0612 = AA18

mentre il contenuto della memoria rimane inalterato.

Il Codice di Condizione assumerà il valore binario 10 in quanto il risultato della sottrazione è > 0.

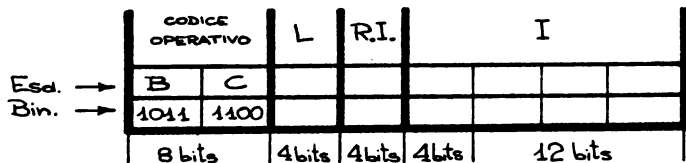
6.3.5 Confronto registro-memoria

Funzione

Questa istruzione è di formato B.
Essa provoca la comparazione binaria fra due dati contenuti rispettivamente in un registro od in una zona di memoria.
I dati sono considerati in formato esadecimale, e le loro cifre esadecimali vengono confrontate successivamente a partire dalla sinistra verso destra.

La lunghezza L esprime il numero di bytes del registro R.I. che vengono confrontati, pertanto essa può assumere soltanto i valori 0 ed 1 (byte di destra), in caso contrario l'istruzione non viene eseguita e la elaborazione si arresta.

L'esecuzione di questa istruzione non modifica i dati ma altera soltanto il valore del codice di condizione.

	CODICE OPERATIVO		L	R.I.	I			
Esa. →	B	C						
Bin. →	1011	1100						
	8 bits		4bits	4bits	4bits	12 bits		

Codice operativo = BC　　　　Codice simbolico = CRM

L = lunghezza, in bytes, diminuita di 1 del contenuto del registro che viene confrontato

R.I. = nome (0÷F) del registro indice interessato nel confronto

I = indirizzo (espresso in modo indiretto) del dato confrontato con il contenuto di R.I.

Fig. 6.6 - Formato del 'Confronto registro-memoria'

Codice Condizione

I valori binari assunti dal 'codice di condizione' hanno i seguenti significati:

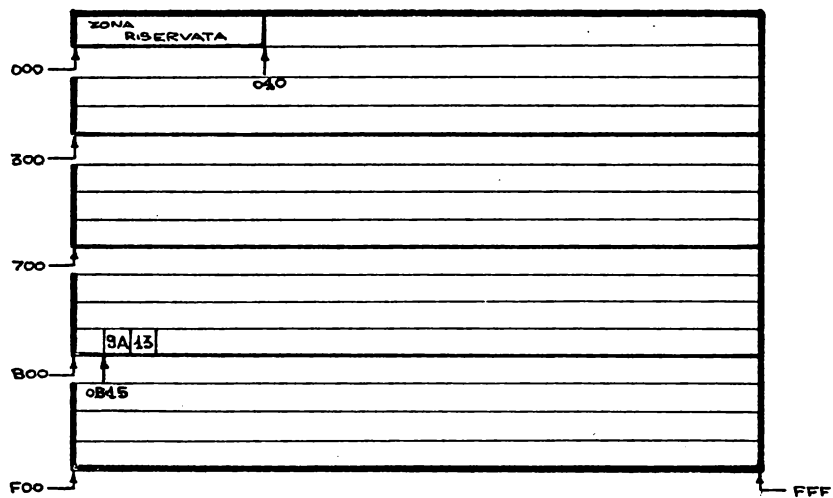
00	se il contenuto di R.I. è uguale al dato in memoria
01	se il contenuto di R.I. è minore del dato in memoria
10	se il contenuto di R.I. è maggiore del dato in memoria

Il valore 11 non può essere assunto dal 'codice di condizione' dopo un confronto registro-memoria.

Esempio

Si consideri la seguente istruzione:

Se il registro 3 contiene il valore 9A12 ed il registro A contiene il valore 0B00, essa provoca il confronto binario (esadecimale) del dato esadecimale 9A12 con il contenuto della zona di memoria di indirizzo 0B00 + 0015 = 0B15.

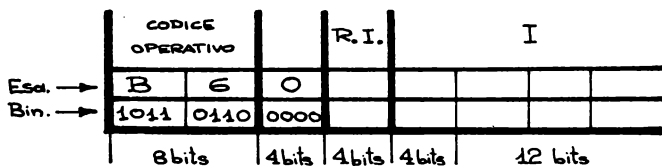
Se la mappa della memoria prima dell'esecuzione della istruzione è la seguente:

Il risultato di tale confronto è 'minore'.
Il codice di condizione assume quindi il valore 01.

6.3.6 Modifica della istruzione SME

L'istruzione SME (salto con memorizzazione) opera in questa estensione del calcolatore CND in un modo leggermente diverso rispetto a quello descritto nel paragrafo 3.3.2.

La differenza di funzionamento di tale istruzione consiste nella scrittura in un qualsiasi registro, indicato nel campo R.I., del valore dell'indirizzo dell'istruzione ad essa successiva anzichè sempre nelle posizioni di indirizzo 0000 e 0001.

Tale modifica permette di rendere più agevole il collegamento tra programma principale e vari sottoprogrammi appartenenti a diversi livelli (paragrafo 6.4.2).

	CODICE OPERATIVO			R.I.	I			
Esa. →	B	6	0					
Bin. →	1011	0110	0000					
	8bits		4bits	4bits	4bits	12 bits		

Campo operativo = B6   Campo simbolico = SME

R.I. = nome (0÷F) del registro indice nel quale viene memorizzato l'indirizzo dell'istruzione successiva alla SME

I = indirizzo (espresso in modo indiretto) dell'istruzione da eseguire

Fig. 6.7 - Formato del 'Salto con memorizzazione'

Codice Condizione

Il codice di condizione non viene gestito da questa istruzione.

Esempio

La seguente istruzione:

provoca il 'salto incondizionato' all'istruzione di indirizzo 030B così ottenuto:

020A + contenuto registro 2
0101
———
030B

e memorizza nel registro indice 'D' l'indirizzo della istruzione sequenzialmente successiva ad essa nel programma.

6.4 USO DEI REGISTRI INDICE NEI SOTTOPROGRAMMI

6.4.1 Rientro e passaggio dei parametri

Con la introduzione dei registri indice i problemi descritti nel Capitolo 5, relativi alla utilizzazione dei sottoprogrammi, vengono risolti in modo decisamente più semplice e più lineare.

Il meccanismo del 'rientro' da un sottoprogramma viene ora realizzato nel seguente modo:

- l'indirizzo della istruzione successiva a quella che effettua il 'richiamo' del sottoprogramma (l'istruzione SME) viene automaticamente memorizzato nel registro R.I. indicato nella istruzione stessa.

- tale registro viene utilizzato dall'ultima istruzione del sottoprogramma (che sarà sempre un 'salto incondizionato') per determinare l'indirizzo di rientro.

Nella figura 6.8 è riportato un esempio di richiamo di un sottoprogramma e di rientro al programma principale.

In tale esempio il registro 6 è stato utilizzato per l'indirizzamento e l'indirizzo di inizio del sottoprogramma è 62A4, e cioè il contenuto del registro 6 aumentato di 2A4.

La esecuzione dell'istruzione SME, che effettua il richiamo, provoca la registrazione automatica nel registro A del valore 6084, che è l'indirizzo di rientro del sottoprogramma.

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				OSSERVAZIONI
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	6	0	8	0	S	M	E	B	6	0	A	6	2	A	4					RICHIAMO DEL SOTTOPROGRAMMA
	6	0	8	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	6	2	A	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	INIZIO DEL SOTTOPROGRAMMA
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
	6	3	1	6	S	A	L	B	5	F	0	A	0	0	0					ULTIMA ISTRUZIONE DEL SOTTOPROGRAMMA

Fig. 6.8 - Richiamo e rientro di un sottoprogramma

L'esecuzione dell'istruzione SAL che chiude il sottoprogramma provoca il rientro al programma principale nella esatta posizione (cioè all'indirizzo 6084) contenuto nel registro A.

Occorre tenere presente che il contenuto del registro R.I. indicato nella SME viene automaticamente alterato dal richiamo del sottoprogramma, e quindi il suo utilizzo per altri scopi (indirizzamento, area di servizio, deposito, ecc.) è decisamente sconsigliabile.

In ogni caso, quando il suo utilizzo è indispensabile, e possibile salvarne il contenuto trasferendolo, prima dell'esecuzione della relativa istruzione SME, in opportuni depositi della memoria, dai quali deve essere prelevato e trasferito nuovamente nel registro dopo il 'rientro' dal sottoprogramma.

Anche le comunicazioni tra programma principale e sottoprogramma (e cioè il passaggio dei parametri in ingresso e in uscita) possono avvenire mediante l'utilizzo dei registri indice.

I valori dei dati che devono essere trasferiti ad un sottoprogramma, o meglio i loro indirizzi, vengono memorizzati dal programma principale direttamente nei registri indice, dai quali verranno prelevati dal sottoprogramma stesso.

Mediante i registri indice viene realizzato anche il passaggio al programma principale dei valori dei parametri in uscita (risultati) di un sottoprogramma.

Nell'esempio descritto nel precedente paragrafo 5.5 lo scambio dei dati tra programma principale e sottoprogramma può essere realizzato utilizzando un solo registro indice, nel quale viene registrato prima di ogni richiamo il parametro di ingresso (numero da quadruplicare), e dal quale dovrà essere prelevato, dopo ogni rientro, il risultato (valore quadruplicato).

Si consideri ora il seguente esempio di sottoprogramma che effettua la moltiplicazione di due numeri decimali interi non segnati aventi al massimo 6 cifre.

L'istruzione SME del programma principale che richiama tale sottoprogramma è la seguente:

B6 OC 5000

Per la sua realizzazione sono state fatte le seguenti convenzioni:

a - i due operandi devono avere sempre 6 caratteri numerici ciascuno; qualora il numero dei caratteri fosse minore di 6 essi devono essere completati a sinistra con zeri non significativi a cura del programma principale

b - il programma principale, prima del richiamo di tale sottoprogramma, deve memorizzare in due registri indice gli indirizzi dei due operandi e precisamente nel registro 1 l'indirizzo del moltiplicatore e nel registro 2 l'indirizzo del moltiplicando

c - il sottoprogramma fornisce il risultato della moltiplicazione in un deposito, detto Accumulatore, il cui indirizzo viene memorizzato nel registro 3

Il flow-chart che illustra l'algoritmo utilizzato per effettuare la moltiplicazione è riportato nella figura 6.9.

I caratteri del moltiplicatore vengono considerati dal sottoprogramma nel seguente ordine:

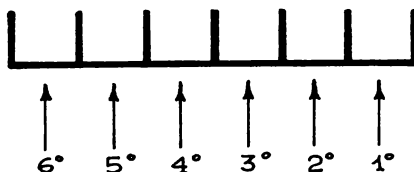
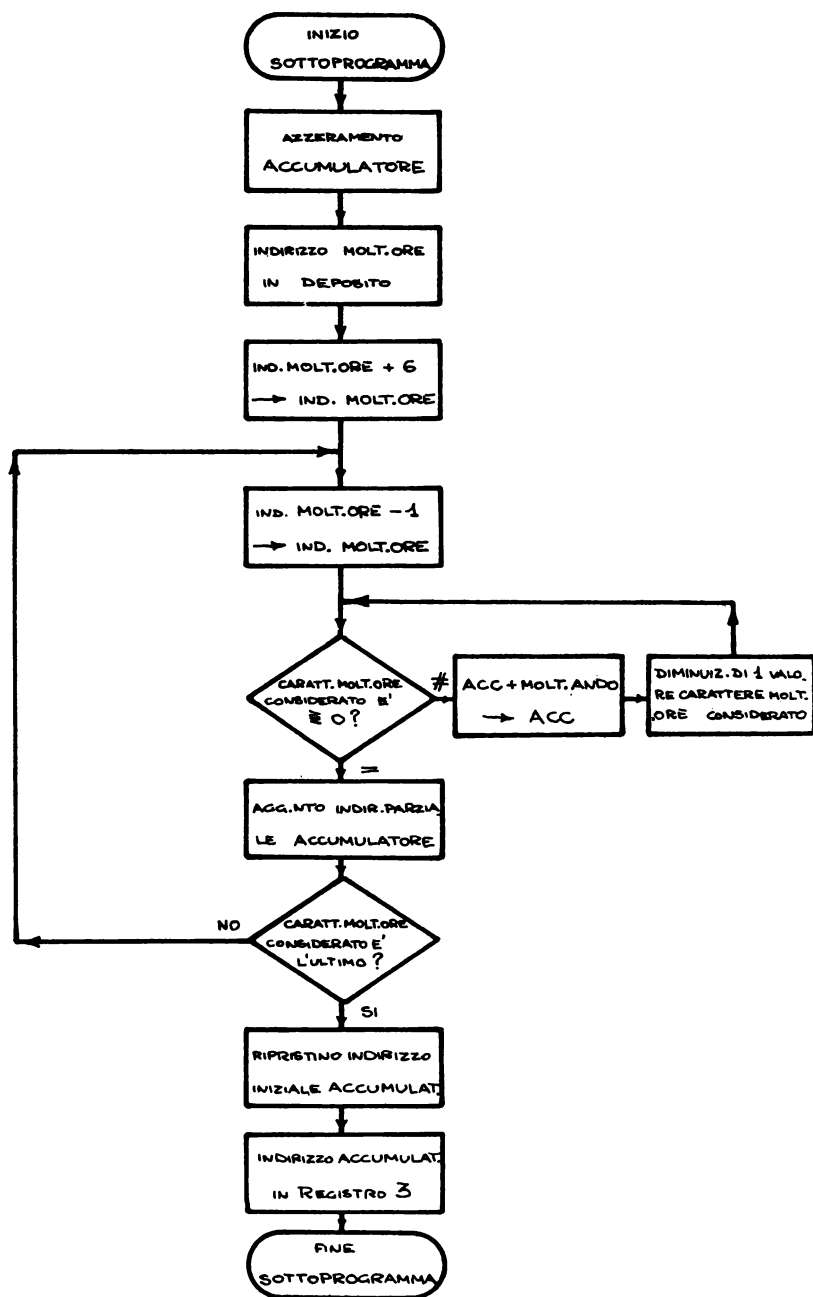



Fig. 6.9 - Sottoprogramma di moltiplicazione

La codifica delle istruzioni del sottoprogramma è riportata nelle figure 6.10 e 6.11 mentre nella figura 6.12 sono riportati i valori delle costanti e dei depositi utilizzati.

Come esempio, si immagini di richiamare la esecuzione del sottoprogramma per effettuare la moltiplicazione dei seguenti operandi:

000523 moltiplicando

000031 moltiplicatore

La sequenza operativa del sottoprogramma che realizza il prodotto può essere schematizzata nel modo seguente:

si considera 1° caratt. molt. ore — 1
valore carattere è = 0 ? — NO
si esegue la somma — ACC + molt.ando → ACC

```
000000000000 +   ACC
      ↑000523 =  molt.ando
-------------
000000000523     ACC
```

1° caratt. molt.ore -1 → 1° caratt. molt.ore — 1 - 1 = 0
valore carattere è = 0 ? — SI
diminuzione indirizzo parziale ACC di un carattere

si considera 2° caratt. molt.ore — 3
valore carattere è = 0 ? — NO
si esegue la somma — ACC + molt.ando → ACC

```
000000000523 +   ACC
     ↑000523 =   molt.ando
-------------
000000005753     ACC
```

SALTI	INDIRIZZO ISTRUZIONE	CODICE SIMBOL.	ISTRUZIONE COD. OPER.	LUNG.	1° OPERANDO	2° OPERANDO	OSSERVAZIONI
	1 2 3 4	5 6 7	8 9	10 11	12 13 14 15	16 17 18 19	
	5000	SDE	A2	BB	5100	5100	AZZERAMENTO ACCUMULATORE
	5006	RME	B9	12	5028		REGISTRAZ. INDIR. MOLT.ANDO NELL'ISTRUZIONE 5024
	500A	RME	B9	11	510E		REGISTRAZ. INDIR. MOLT.ORE IN DEPOSITO
	500E	ARM	BA	11	510D		INDIR. MOLT.ORE + 6 → REGISTRO 1
	5012	SRM	BB	11	5110		INDIR. MOLT.ORE −1 → INDIR. MOLT.ORE
	5016	RME	B9	11	501C		COMPILAZIONE ISTRUZIONE SUCCESSIVA
	501A	CAL	A7	00		510C	IL CARATTERE DEL MOLT.ORE CONSIDERATO È ≧ 0?
	5020	SAL	B5	80	5038		SE IL CARATTERE CONSIDER. È ≥ 0 SI SALTA ALL'IST. 5038
	5024	ADE	A1	65	5105		ACCUMULATORE + MOLT.ANDO → ACCUMULATORE
	502A	RME	B9	11	5030		COMPILAZIONE ISTRUZIONE SUCCESSIVA
	502E	SDE	A2	00		5111	DIMINUZIONE DI 1 UNITÀ DEL VALORE DEL CARATTERE CONSID
	5034	SAL	B5	F0	501A		SALTO INCONDIZIONATO AL CONFRONTO CON 0
	5038	SES	A4	31	5026	5110	POSIZIONAMENTO A SINISTRA DELL'INDIRIZZO PARZ. ACCUM
	503E	CRM	BC	11	510E		IL CARATTERE DEL MOLT.ORE CONSIDERATO È L'ULTIMO?
	5042	SAL	B5	70	5012		SE NON È L'ULTIMO SALTO ALL'ISTRUZIONE 5012
	5046	MRE	B8	13	5026		REGISTRAZ. INDIRIZZO ACCUMULATORE IN REG. 3
	504A	AES	A3	31	5026	510D	RIPRISTINO INDIRIZZO INIZIALE ACCUMULATORE
	5050	ARM	BA	13	5110		CALCOLO VALORE ESATTO INDIRIZZO ACCUMULATORE

Fig. 6.10 - Sottoprogramma di moltiplicazione

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	5	0	5	4	S	A	L	B	5	F	0	C	0	0	0					ULTIMA ISTRUZ. SOTTOPROGRAMMA - ISTRUZ. DI RIENTRO

Fig. 6.11 - Sottoprogramma di moltiplicazione (cont.)

SALTI	INDIRIZZO ISTRUZIONE				CODICE SIMBOL.			ISTRUZIONE												OSSERVAZIONI
								COD. OPER.		LUNG.		1° OPERANDO				2° OPERANDO				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	5	1	0	0				6	0	6	0	6	0	6	0	6	0	6	0	ACCUMULATORE - 12 bytes
	5	1	0	6				6	0	6	0	6	0	6	0	6	0	6	0	
	5	1	0	C				6	0											COSTANTE DECIMALE '0'
	5	1	0	D				0	6											COSTANTE ESADECIMALE '6'
	5	1	0	E				X	X	X	X									DEPOSITO PER INDIRIZZO MOLTIPLICATORE - 2 bytes
	5	1	1	0				0	1											COSTANTE ESADECIMALE '1'
	5	1	1	1				6	1											COSTANTE DECIMALE '1'

Fig. 6.12 - Costanti e depositi

2° caratt. molt.ore -1 ➝ 2° caratt.molt.ore   3 - 1 = 2
valore carattere è = 0 ?   NO
si esegue la somma   ACC + molt.ando ➝ ACC

```
000000005753 +   ACC
   ↑000523   =   molt.ando
-------------
000000010983     ACC
```

2° caratt. molt.ore -1 ➝ 2° caratt.molt.ore   2 - 1 = 1
valore carattere è = 0 ?   NO
si esegue la somma   ACC + molt.ando ➝ ACC

```
000000010983 +   ACC
   ↑000523   =   molt.ando
-------------
000000016213     ACC
```

2° caratt. molt.ore -1 ➝ 2° caratt.molt.ore   0
valore carattere è = 0 ?   SI
diminuire indirizzo parziale ACC di un carattere

Tali operazioni si ripetono sino a che non sono stati considerati tutti i caratteri del moltiplicatore.
Alla fine il risultato esatto della moltiplicazione si trova nell'Accumulatore.
Il sottoprogramma considerato può facilmente essere modificato per poter effettuare la moltiplicazione di due operandi aventi lunghezze variabili.
In tale caso le lunghezze dei due operandi dovranno essere passate ogni volta al sottoprogramma come parametri.

6.4.2 Livelli di sottoprogrammi

In ogni programma principale possono essere richiamati più sottoprogrammi (SP1, SP2, SP3, ..., SPN).
Se i sottoprogrammi sono tutti indipendenti tra di loro, essi verranno detti di livello 1 ed il loro richiamo viene effettuato sempre dal programma principale (figura 6.13).

Il collegamento tra il programma principale ed i vari sottoprogrammi di Livello 1 può essere realizzato utilizzando lo stesso registro indice, in quanto il suo contenuto viene alterato soltanto dalla successiva istruzione SME (e cioè quando l'esecu - zione del sottoprogramma precedente è già terminata ed è già stato effettuato il rientro).

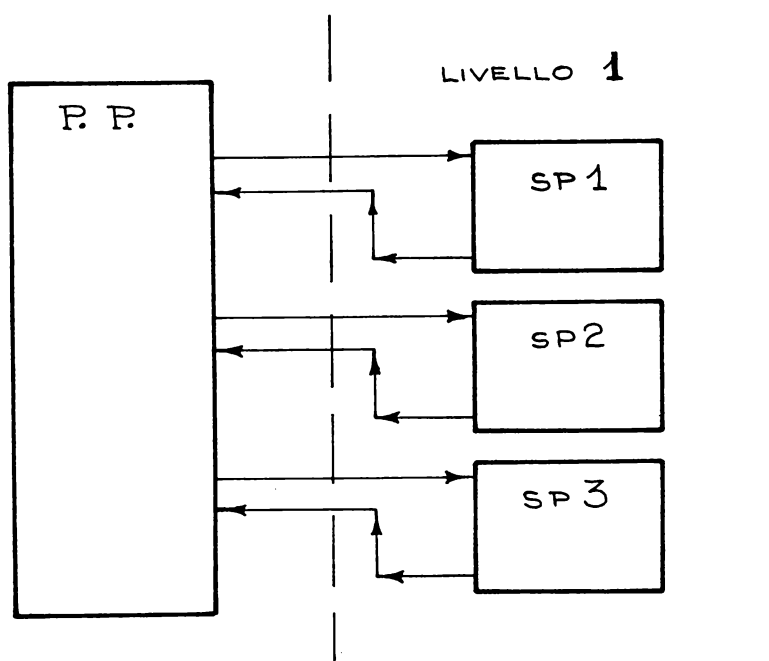



Fig. 6.13 - Sottoprogrammi di Livello 1

Se i sottoprogrammi non sono indipendenti tra loro, nel senso che un sottoprogramma è richiamato da un altro sottoprogramma oppure è richiamato sia dal programma principale che da un altro sottoprogramma, non è più possibile utilizzare lo stesso registro indice in quanto il suo contenuto verrebbe alterato prima del termine dell'esecuzione del sottoprogramma precedente, con la conseguente perdita dell'esatto indirizzo di rientro.

In questo caso è opportuno definire diversi livelli di sottoprogrammi ed associare ad ogni livello un proprio registro indice. Un sottoprogramma appartiene al Livello N se è richiamato da almeno un sottoprogramma di Livello N-1 (figura 6.14).
Per il richiamo di ogni sottoprogramma, in funzione del suo livello di appartenenza, deve essere utilizzato il particolare registro indice associato a quel livello.

Si immagini ad esempio di utilizzare per il richiamo dei sottoprogrammi di Livello N il registro N, e cioè il registro indice corrispondente al numero del livello, e si consideri il seguente esempio di richiamo di sottoprogrammi di due livelli (1 e 2).

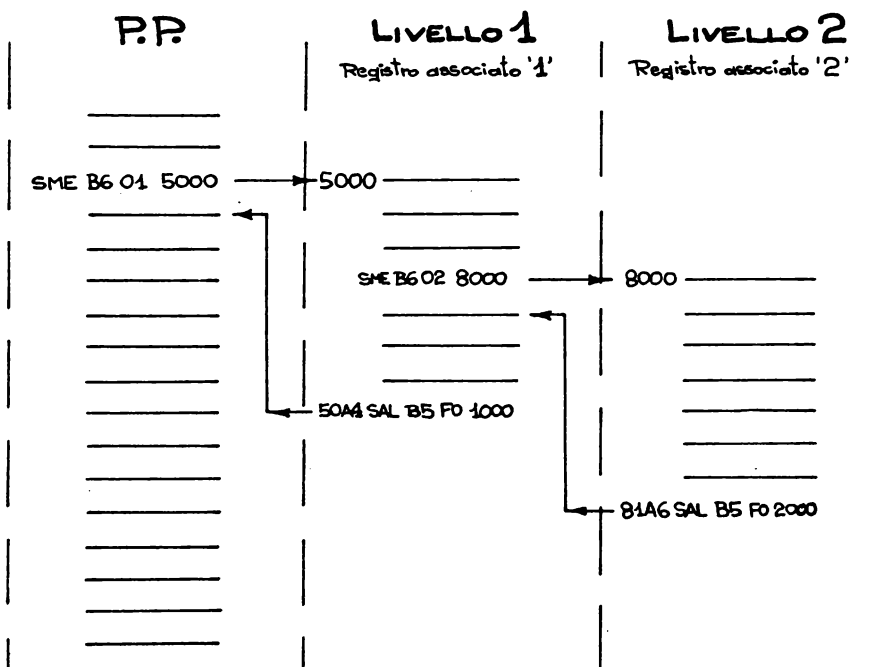
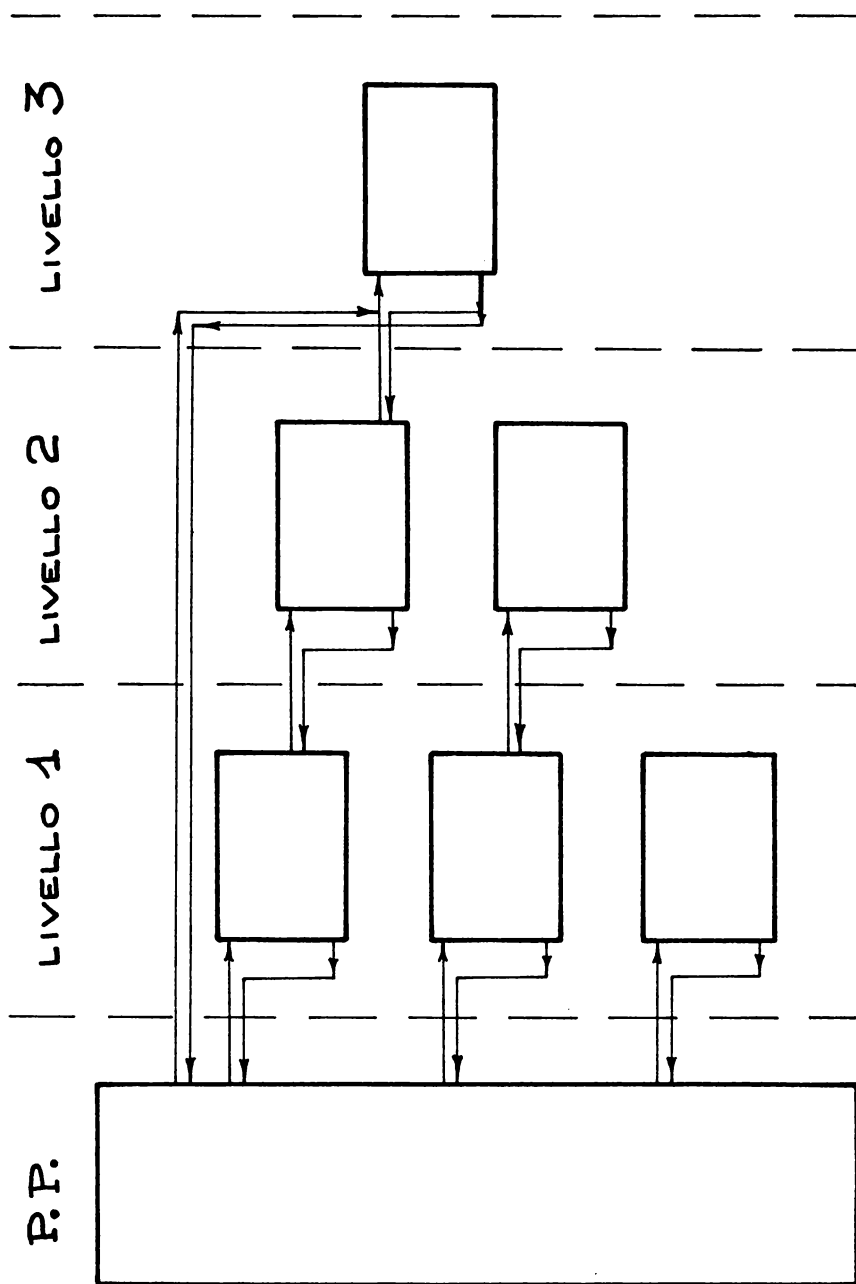



Fig. 6.14 - Sottoprogrammi a diversi livelli

Occorre tenere presente che il contenuto dei vari registri indice utilizzati per i collegamenti dei sottoprogrammi di diversi livelli non deve mai essere alterato sino a che la esecuzione dei sottoprogrammi di quella particolare catena non è terminata.

6.5 RICOLLOCABILITA' DEI PROGRAMMI

Uno dei principali vantaggi ottenuti dall'introduzione dei 'registri indice' nel calcolatore CND è quello della 'ricollocabilità' dei programmi.

Un programma si dice 'ricollocabile' quando può essere caricato ed eseguito in qualsiasi posizione della memoria senza dover subire alcuna modifica.

Se si considera un programma che è stato codificato utilizzando il metodo di indirizzamento diretto (e cioè senza l'uso dei registri indice), appare subito evidente che esso deve sempre essere caricato ed eseguito nelle stesse posizioni di memoria i cui valori compaiono direttamente nelle istruzioni.

Questa limitazione non esiste in un programma che è stato codificato utilizzando il metodo di indirizzamento indiretto; in esso infatti tutti gli indirizzi (delle istruzioni, delle costanti, delle aree di lavoro, dei depositi) sono relativi al valore contenuto nel registro indice utilizzato come riferimento per tutto il programma.

Quindi, con tale metodo di indirizzamento, un programma può essere caricato ed eseguito in posizioni diverse di memoria purchè nel registro indice usato nelle istruzioni come riferimento sia caricato un opportuno valore.

Una utilizzazione pratica della 'ricollocabilità' dei programmi si ha quando nella memoria del calcolatore si trovano registrati contemporaneamente più programmi.

Si consideri il seguente esempio:

in un calcolatore CND avente la capacità di memoria di 4K (4096 bytes) si debbano eseguire alternativamente i seguenti due programmi:

1° gruppo
- programma PRIMO (lunghezza = 960 bytes)
- programma SECONDO (lunghezza = 2 K)

2° gruppo
- programma TERZO (lunghezza = 2 K)
- programma SECONDO (lunghezza = 2 K)

Le mappe della occupazione di memoria nei due casi sono riportate nelle figure 6.15 e 6.16.

Durante l'esecuzione dei programmi del 1° gruppo i registri indice degli indirizzi dei programmi PRIMO e SECONDO contengono rispettivamente i valori:

0040 0400

Poichè il programma SECONDO deve essere eseguito, nel caso del 2° gruppo, in una nuova posizione di memoria, il contenuto del suo registro indice degli indirizzi deve essere modificato, e precisamente deve assumere il valore:

0800

Con questo nuovo valore del registro indice, il programma SECONDO, potrà essere caricato ed eseguito senza la necessità di apportare alcuna modifica al suo testo.

Quando deve essere eseguito nuovamente il 1° gruppo di programmi, basterà ricaricare nel registro indice degli indirizzi del programma SECONDO il valore

0400

La 'ricollocabilità' viene principalmente utilizzata nei grossi calcolatori nei quali esiste la possibilità della esecuzione contemporanea di più programmi, tutti registrati nella memoria (MULTIPROGRAMMAZIONE).

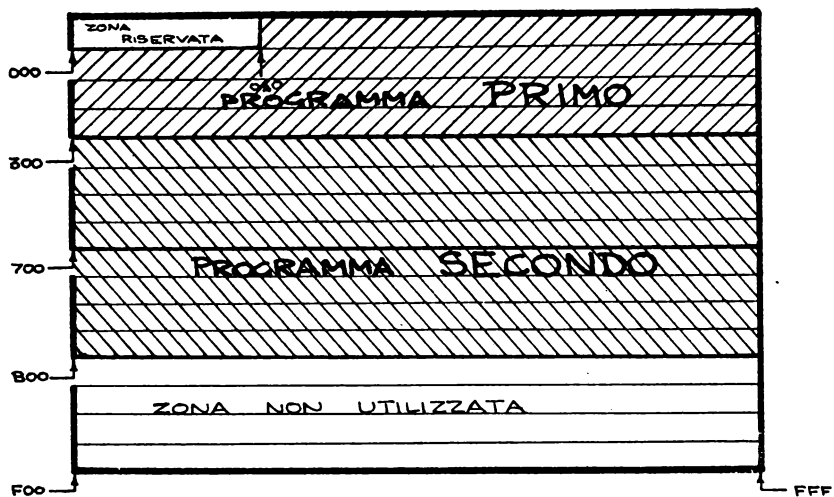



Fig. 6.15 - Occupazione della memoria nel 1° caso

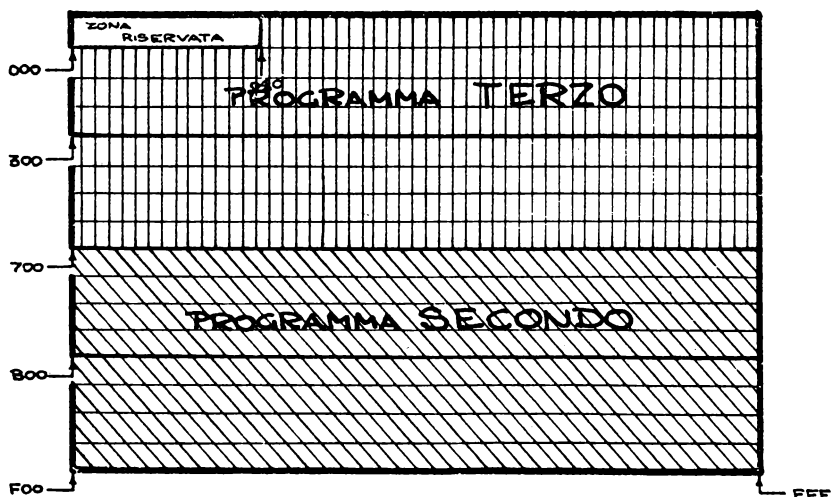



Fig. 6.16 - Occupazione della memoria nel 2° caso

CAPITOLO 7

IL LINGUAGGIO ASSEMBLATIVO C.N.D.

7.1 INTRODUZIONE

7.2 ISTRUZIONI DEL LINGUAGGIO ASSEMBLATIVO C.N.D.

7.3 PROGRAMMI IN LINGUAGGIO ASSEMBLATIVO

7.1 INTRODUZIONE

La programmazione in linguaggio base di un calcolatore comporta senza dubbio notevoli difficoltà delle quali le principali sono:

a - l'uso di codici numerici per definire il tipo di operazione e la necessità di esprimere mediante valori esadecimali gli indirizzi degli operandi, possono facilmente dare origine a banali errori di scrittura

b - l'assegnazione degli indirizzi sia dei dati che delle istruzioni del programma può essere facilmente fonte di errore, in quanto richiede la esecuzione manuale di operazioni aritmetiche di tipo esadecimale

c - dopo avere completato la codifica di un programma può presentarsi la necessità di apportarvi delle modifiche, sia per eliminare gli errori di scrittura, sia per ottenere prestazioni migliori o comunque diverse da quelle inizialmente previste. Queste necessità, per essere soddisfatte, possono causare frequentemente la riscrittura dell'intero programma o di una notevole parte di esso. Infatti l'inserimento di nuove istruzioni in una sequenza preesistente provoca un incremento dei valori degli indirizzi di tutte le istruzioni successive, con la conseguente necessità di modificare gli indirizzi di salto e gli indirizzi delle costanti e delle aree di memoria utilizzate nel programma.

La redazione di programmi in linguaggio macchina richiede quindi da parte del programmatore una conoscenza estremamente approfondita del calcolatore ed un lungo periodo di addestramento.

Le difficoltà ora elencate od altre che dipendono dal particolare tipo di calcolatore utilizzato sono tutte collegabili a ben precise fasi del lavoro di codifica in linguaggio macchina, e cioè alla scrittura dei codici operativi, alla assegnazione degli indirizzi delle costanti ed aree di lavoro, al calcolo degli

indirizzi delle istruzioni, fasi che sono tutte riducibili ad operazioni di tipo meccanico.

Quindi sorge spontanea l'idea di affidare al calcolatore stesso, particolarmente adatto a trattare problemi facilmente schematizzabili, il lavoro di traduzione dei codici operativi che il programmatore indicherà mediante codici simbolici ed il lavoro di assegnazione degli indirizzi alle istruzioni, alle costanti e ai depositi che verranno individuati mediante nomi simbolici.

Lo sviluppo di questa idea ha portato alla definizione di nuovi linguaggi di programmazione, chiamati 'linguaggi simbolici', a ciascuno dei quali è connesso un 'programma traduttore'.

Il programma scritto utilizzando un linguaggio simbolico viene denominato 'programma sorgente' ed il corrispondente programma tradotto in linguaggio macchina viene denominato 'programma oggetto'.

Un linguaggio simbolico si dice 'orientato alla macchina' quando la sua sintassi è strettamente legata alle caratteristiche del calcolatori per il quale è definito.

Un linguaggio simbolico 'orientato alla macchina' viene spesso chiamato linguaggio 'uno a uno' per il fatto che ad ogni istruzione macchina corrisponde una istruzione del linguaggio simbolico.

Tale corrispondenza non è in realtà sempre 'uno a uno' in quanto il linguaggio simbolico comprende anche istruzioni che non hanno corrispondenza tra le istruzioni del linguaggio macchina.

Alcune di tali istruzioni possono infatti contenere informazioni necessarie al programma traduttore, mentre altre possono dare origine, durante la traduzione, a più istruzioni in linguaggio macchina o provocare l'incorporazione nel programma oggetto di un intero sottoprogramma.

I programmi che traducono programmi sorgente redatti in linguaggio simbolico orientato alla macchina nel corrispondente programma 'oggetto' in linguaggio macchina vengono comunemente chiamati

'ASSEMBLATORI'.
La fase di traduzione di un programma sorgente viene perciò chia
mata 'Assemblaggio' del programma, ed il linguaggio sorgente vie
ne chiamato 'linguaggio assemblativo'.

7.2 ISTRUZIONI DEL LINGUAGGIO ASSEMBLATIVO C.N.D.

7.2.1 Generalità

Le istruzioni del linguaggio assemblativo del calcolatore CND possono essere classificate nei seguenti gruppi:

- istruzioni esecutive
- istruzioni dichiarative

Le istruzioni esecutive vengono utilizzate per eseguire le operazioni richieste dal problema ed è possibile porle in corrispondenza biunivoca con le istruzioni del linguaggio macchina descritte in precedenza (capitolo 3).

Le istruzioni dichiarative vengono utilizzate per descrivere le costanti, e cioè i dati ai quali viene inizialmente assegnato un valore prestabilito, oppure le aree di memoria riservate per contenere dati di valore non prefissato.

Alle istruzioni di tale gruppo non corrisponde alcuna istruzione nel linguaggio macchina.

7.2.2 Formato delle istruzioni esecutive

Le istruzioni del linguaggio macchina del calcolatore CND descritte nei capitoli 3 e 6 (7 istruzioni di formato A ed 11 istruzioni di formato B) sono le corrispondenti nel linguaggio macchina di tutte istruzioni esecutive del linguaggio assemblatore CND.

Le 'istruzioni esecutive' corrispondenti alle istruzioni macchina di formato A hanno il seguente formato:

ETICHETTA	COD.SIMBOLICO	LUNG.	1°OPERANDO, 2°OPERANDO

mentre quelle corrispondenti alle istruzioni oggetto di formato B hanno il seguente formato:

ETICHETTA	COD.SIMBOLICO	LUNG.	OPERANDO

La descrizione dei singoli campi è la seguente:

ETICHETTA

E' costituita da una combinazione di caratteri alfabetici e numerici non superiore a 4, il primo dei quali deve essere sempre alfabetico.
Essa serve per l'identificazione di una istruzione nell'ambito del programma. L'identificazione delle istruzioni viene così effettuata mediante le etichette (cioè con un nome simbolico) e non più mediante i valori esadecimali degli indirizzi delle istruzioni stesse.
Sono esempi di etichette scritte in modo corretto i seguenti:

A866
BEPP
SAL1
SE
SI
NO
B
C8A2

Le seguenti etichette sono invece errate:

1EE - (il 1° carattere è numerico)
E(5) - (le parentesi non sono ammesse)
SAL12- (è costituita da 5 caratteri)

Un nome simbolico deve comparire una sola volta nel campo 'Etichetta' di un programma, ma può comparire più volte nel campo 'operandi'.
Per evitare inutili complicazioni è bene utilizzare le 'etichette' soltanto dove la loro presenza è indispensabile per motivi di programmazione.

COD.SIMBOLICO

E' costituito da un simbolo di 3 caratteri alfabetici.
Tali simboli servono per distinguere in modo univoco le varie istruzioni esecutive, e sono in corrispondenza biunivoca con i 'codici operativi' delle istruzioni del linguaggio macchina.
Tutti i codici simbolici delle istruzioni esecutive del linguaggio assemblativo sono elencati nella figura 7.1.

COD. SIMB.	FUNZIONE
ADE	addizione decimale
SDE	sottrazione decimale
AES	addizione esadecimale
SES	sottrazione esadecimale
TRA	trasferimento
CLO	confronto logico
CAL	confronto algebrico
IES	introduzione esadecimale
IAL	introduzione alfanumerica
OES	estrazione esadecimale
OAL	estrazione alfanumerica
SAL	salto
ALT	arresto
SME	salto con memorizzazione
MRE	trasferimento memoria-registro
RME	trasferimento registro-memoria
ARM	addizione registro-memoria
SRM	sottrazione registro-memoria
CRM	confronto registro-memoria

Fig. 7.1 - Codici Simbolici delle istruzioni esecutive

LUNGHEZZA

E' costituita da due cifre esadecimali che, in funzione della particolare istruzione considerata, assumono i significati già descritti nel Capitolo 3 e nel Capitolo 6 per il 2° byte delle istruzioni oggetto. La lunghezza può anche essere omessa; in tal caso gli operandi vengono automaticamente considerati di lunghezze uguali a quelle definite nelle relative 'istruzioni dichiarative'.

OPERANDI

Sono due nomi simbolici che rappresentano indirizzi di costanti, di aree di lavoro o di istruzioni.
Essi devono essere costituiti da una qualsiasi combinazione di caratteri alfanumerici secondo le regole già descritte per la 'etichetta'. In tutte le istruzioni esecutive corrispondenti alle istruzioni di formato A, nelle quali è necessaria la presenza di entrambi i simboli, i due operandi devono essere separati mediante il carattere ',' (virgola).
Naturalmente, se un simbolo compare almeno una volta nel campo 'operandi', dovrà sicuramente comparire anche nel campo 'etichetta' di una istruzione dello stesso programma.
Per dare maggiore elasticità al linguaggio, gli operandi possono anche essere espressi nella forma 'registro indice + indirizzo relativo', e cioè con 4 cifre esadecimali come si è già visto nel Capitolo 6.

7.2.3 Formato delle istruzioni dichiarative

Le istruzioni dichiarative del linguaggio assemblativo possono essere raggruppate nelle seguenti due classi in funzione delle operazioni da esse svolte:

(a) - descrizione delle costanti

(b) - definizione delle lunghezze delle aree di lavoro

Le istruzioni che svolgono le funzioni descritte nel punto (a) sono due, in quanto la costante da esse definita può essere di tipo alfanumerico o di tipo esadecimale.

Il formato di tali istruzioni è il seguente:

ETICHETTA	COD.SIMBOLICO	'valore della costante'

La descrizione dei singoli campi è la seguente:

ETICHETTA

E' un nome simbolico che deve sempre essere presente.
Il programmatore deve utilizzare nel programma tale nome simbolico per accedere alla costante descritta nella istruzione stessa. Le regole per la composizione di tale nome simbolico sono quelle già descritte in precedenza per le istruzioni esecutive.

COD.SIMBOLICO

E' costituito da un simbolo di 3 caratteri e serve per specificare il tipo di dichiarazione che si intende fare, e cioè se la costante è di tipo alfanumerico (1 byte per ogni carattere del valore della costante) oppure di tipo esadecimale (1 semibyte per ogni carattere del valore della costante).
I codici simbolici possibili sono i seguenti:

COE - costante esadecimale
COA - costante alfanumerica

'valore della costante'

Il valore della costante o del deposito viene descritto in questo campo e deve essere racchiuso tra due apici.
Nel caso di una istruzione COE le cifre esadecimali che compaiono all'interno degli apici devono essere in numero pari.
La lunghezza di una costante alfanumerica (COA) viene calcolata automaticamente dal calcolatore considerando il numero dei caratteri racchiusi tra gli apici, mentre quella di una

costante esadecimale (COE) dimezzando tale numero. Se il testo di una costante alfanumerica comprende il carattere ' (apice), è necessario ripeterlo consecutivamente due volte, in modo da distinguerlo dal carattere che segna la fine del testo della costante stessa.

Si considerino, ad esempio, le seguenti istruzioni dichiarative:

La rappresentazione in memoria delle costanti descritte da tali due istruzioni sarà rispettivamente:

1	2

6	1	6	2

Come si può notare, la prima costante è lunga 1 byte (2 cifre esadecimali), mentre la seconda è lunga 2 bytes (2 caratteri alfanumerici).

Il formato delle istruzioni dichiarative che servono per descrivere le aree di memoria è il seguente:

ETICHETTA	COD.SIMBOLICO	lunghezza

La descrizione dei singoli campi è la seguente:

ETICHETTA

E' il nome simbolico dell'area di memoria considerata, e deve sempre essere presente in ogni istruzione di tale tipo.

COD.SIMBOLICO

E' costituito da un simbolo di 3 caratteri e serve per specificare la dichiarazione che si intende fare.

Esso è:

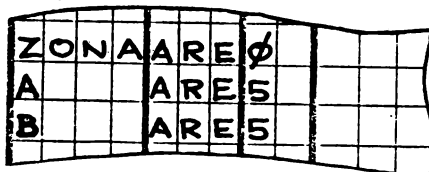
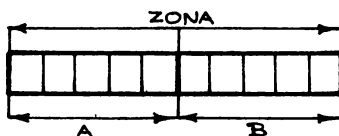
ARE - definizione di area

lunghezza

E' costituito da un numero decimale, avente al massimo 3 caratteri, che specifica la lunghezza (in bytes) dell'area che viene riservata dalla istruzione considerata.
Se la lunghezza specificata è zero, l'area riservata è di ampiezza nulla il che consente di associare a questa area e a quella immediatamente successiva lo stesso indirizzo (possono invece essere diverse le lunghezze). Ciò permette di definire dei campi all'interno di un'area.

Ad esempio la seguente istruzione:

definisce un'area di memoria di nome BA13 e di lunghezza 18 bytes.

Le seguenti istruzioni:

permettono di definire due campi lunghi 5 bytes ciascuno all'interno dell'area di nome ZONA.

Va tenuto presente che la lunghezza associata al nome simbolico ZONA è nulla e che pertanto in eventuali frasi esecutive aventi ZONA come operando sarà necessario esprimere esplicitamente la lunghezza effettiva.

Le istruzioni dichiarative possono trovarsi all'inizio o alla fine del programma sorgente, ma possono anche essere mescolate ad istruzioni esecutive.
In ogni caso il programmatore dovrà tener conto che ad esse non corrisponde una istruzione macchina eseguibile.

L'assemblatore, per potere stabilire quando le istruzioni costituenti un programma sono terminate, ha la necessità di incontrare una apposita istruzione dichiarativa.
Tale istruzione dichiarativa è costituita soltanto dal codice operativo simbolico END, e deve sempre essere l'ultima di ogni programma.

7.3 PROGRAMMI IN LINGUAGGIO ASSEMBLATIVO

7.3.1 Redazione di un programma

Un programma in linguaggio Assemblativo viene generalmente scritto su appositi moduli, denominati 'fogli di programmazione' (vedi figura 7.2).

Ogni riga del 'foglio di programmazione' è suddivisa in 18 caselle numerate, in ciascuna delle quali può essere scritto un solo simbolo grafico (carattere alfanumerico); in ogni riga può essere scritta una sola istruzione.

Nelle colonne 1, 2, 3, 4 deve essere scritta la eventuale 'ETICHETTA', che serve per la identificazione di quella particolare istruzione.
Se la ETICHETTA ha lunghezza inferiore a 4 caratteri, essa deve essere allineata sulla sinistra del campo (colonna 1) e completata con spazi ($\not{b}$). Occorre ricordare che due istruzioni non possono essere identificate dalla stessa ETICHETTA.

Nella figura 7.3 sono riportati alcuni esempi di istruzioni identificate in modo corretto e di istruzioni identificate in modo errato.

Nelle colonne 5, 6, 7 deve essere scritto il codice simbolico dell'istruzione.

Nelle colonne 8, 9 devono comparire i valori delle lunghezze degli operandi, nel caso si vogliano esprimere esplicitamente, oppure due caratteri $\not{b}$, nel caso si utilizzi l'assegnazione implicita delle lunghezze (cioè le lunghezze degli operandi vengono automaticamente considerate uguali a quelle definite nelle relative istruzioni dichiarative).

Nelle colonne 10÷18 devono essere scritti i due operandi (o l'unico operando) in forma simbolica, oppure in forma esplicita, separati dal carattere , (virgola).

Foglio Programma										- Linguaggio ASSEMBLATIVO C.N.D.								
ETICHETTA				COD.S.			LUN.		OPERANDI									COMMENTI
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

Fig. 7.2 - Foglio di programmazione

Foglio Programma – Linguaggio ASSEMBLATIVO C.N.D.				
ETICHETTA (1-4)	COD.S. (5-7)	LUN. (8-9)	OPERANDI (10-18)	COMMENTI
A1CD	TRA		EA,AE	ETICHETTE CORRETTE
PA	ADE	82	TETT,TATT	
CZA	SES		EA11,B2	
A123	AES		MED,PER	
8A51	CAL		E8,AA	ETICHETTE ERRATE
A8 2	SDE		EE,EF	
BBB	TRA		TELA,ALTE	
C2,A	SES		E9,E9	
(83)	ADE		A5,A82	

Fig. 7.3 - Identificazione corretta ed errata di istruzioni

Poichè il carattere alfabetico 'O' ed il carattere numerico 'O' (zero) possano essere facilmente compresi è opportuno stabilire la convenzione che lo zero viene rappresentato graficamente nel seguente modo: Ø.

7.3.2 Esempi di programmi

ESEMPIO 1

Come primo esempio si consideri nuovamente il problema già trattato nel paragrafo 4.4.1, e lo si voglia risolvere ora utilizzando il linguaggio assemblativo.

Il testo l'analisi del problema ed il flow-chart rimangono inalterati (pagg. 117-118) mentre il nuovo programma in linguaggio assemblativo ed i valori delle costanti e dei depositi sono riportati nelle figure 7.4 e 7.5.

Confrontando i due programmi riportati nelle figure 4.7 e 7.4 si può subito notare come il linguaggio assemblativo sia di più semplice applicazione che non il linguaggio macchina.

Nella prima istruzione del programma di fig. 7.4 compaiono in modo esplicito i valori delle lunghezze dei due operandi; tale dichiarazione esplicita è necessaria poichè la funzione delle istruzioni SDE è quella di azzerare 8 bytes di memoria (depositi DEP1 e DEP2) mentre la lunghezza implicita del deposito DEP1 è di 4 bytes.

I valori delle 'maschere' nelle istruzioni di salto SAL assumono gli stessi significati già descritti nel paragrafo 3.3 (pag. 96).

Foglio Programma – Linguaggio ASSEMBLATIVO C.N.D.				
ETICHETTA (1-4)	COD.S (5-7)	LUN (8-9)	OPERANDI (10-18)	COMMENTI
	SDE	77	DEP1,DEP1	AZZERAMENTO DEPOSITI DEP1 DEP2
INTR	IAL		DEPT	INTRODUZIONE DI DUE CARATTERI NUMERICI
	CAL		DEPT,CS1	CONFRONTO DEL NUMERO INTRODOTTO CON '50'
	SAL	B	SAL1	SE IL RISULTATO E' ≥ SALTO ALL'ISTRUZIONE SAL1
	ADE		DEP2,DEPT	SOMMA DEL NUMERO AL DEPOSITO DEP2
	SAL	F	SAL2	SALTO INCONDIZIONATO ALL'INDIRIZZO SAL2
SAL1	ADE		DEP1,DEPT	SOMMA DEL NUMERO AL DEPOSITO DEP1
SAL2	ADE		I,CS2	SOMMA DI '1' AL DEPOSITO I
	CAL		I,CS3	CONFRONTO DI I CON '35'
	SAL	D	INTR	SE IL RISULTATO E' ≤ SALTO ALL'ISTRUZIONE INTR
	OAL		DEP1	STAMPA DEL CONTENUTO DI DEP1
	OAL		DEP2	STAMPA DEL CONTENUTO DI DEP2
	ALT	ØØ	ØØØ1	FINE DELLA ELABORAZIONE
	END			FINE DEL PROGRAMMA

Fig. 7.4 - Esempio 1 - Programma

Foglio Programma									- Linguaggio ASSEMBLATIVO C.N.D.									
ETICHETTA				COD.S.			LUN.		OPERANDI									COMMENTI
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
D	E	P	T	C	O	A	'	Ø	Ø	'								DEPOSITO DI TRANSITO (VALORE INIZIALE '00')
C	S	1		C	O	A	'	5	Ø	'								COSTANTE DECIMALE '50'
D	E	P	1	C	O	A	'	Ø	Ø	Ø	Ø	'						DEPOSITO DEP1
D	E	P	2	C	O	A	'	Ø	Ø	Ø	Ø	'						DEPOSITO DEP2
C	S	2		C	O	A	'	1	'									COSTANTE DECIMALE '1'
I				C	O	A	'	Ø	Ø	'								DEPOSITO I PER IL NUMERO DEI DATI GIA' TRATTATI
C	S	3		C	O	A	'	3	5	'								COSTANTE DECIMALE '35'

Fig. 7.5 - Esempio 1 - Costanti e depositi

ESEMPIO 2

Testo

Costruire un programma che consenta la introduzione successiva di 30 numeri decimali segnati di 2 cifre e la determinazione del numero di valore minimo.

Alla fine del programma dovrà essere stampato il valore minimo trovato.

Analisi

Ogni numero introdotto deve essere registrato in una stessa zona di memoria, denominata TRAN, la quale svolge la funzione di deposito di transito.

Un secondo deposito, MIN, (di valore iniziale + 99) è destinato a contenere il dato cercato di minimo valore.
Ogni dato, dopo essere stato registrato nel deposito TRAN, viene confrontato con il contenuto del deposito MIN.

Se il risultato di tale confronto è 'minore' il suo valore viene sostituito a quello contenuto in MIN, in caso contrario si considera il dato successivo.

Il deposito MIN ha valore iniziale + 99; questo fatto permette di realizzare il programma con il minimo numero di istruzioni possibile, infatti anche il trattamento del primo dato avviene nel procedimento iterativo che serve per tutti gli altri dati.

Flow-chart

Con la possibilità nel linguaggio assemblativo di assegnare nomi simbolici ai depositi ed alle costanti, anche i commenti all'interno dei simboli grafici dei flow-charts possono essere decisamente più concisi.

Foglio Programma - Linguaggio ASSEMBLATIVO C.N.D.																		
ETICHETTA				COD.S.			LUN.		OPERANDI									COMMENTI
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
U	N	O		C	O	A	'	∅	1	'								COSTANTE DECIMALE '1'
I				A	R	E	2											DEPOSITO CHE CONTIENE IL NUMERO DEI DATI GIA' INTRODOTTI
M	I	N		A	R	E	3											DEPOSITO MIN
P	9	9		C	O	A	'	+	9	9	'							COSTANTE DECIMALE '+99' (VALORE INIZIALE DEPOSITO MIN)
T	R	A	N	A	R	E	3											DEPOSITO DI TRANSITO TRAN
T	N	T		C	O	A	'	3	∅	'								COSTANTE DECIMALE '3∅' (VALORE FINALE DI I)

Fig. 7.6 - Esempio 2 - Costanti e depositi

Foglio Programma - Linguaggio ASSEMBLATIVO C.N.D.

ETICHETTA (1-4)	COD.S. (5-7)	LUN. (8-9)	OPERANDI (10-18)	COMMENTI
	TRA		I,UNO	VALORE INIZIALE DEL DEPOSITO I (I=1)
	TRA		MIN,P99	VALORE INIZIALE DEL DEPOSITO MIN (MIN = +99)
ST1	IAL		TRAN	INTRODUZIONE DI 3 CARATTERI ALFANUMERICI
	CAL		TRAN,MIN	CONFRONTO ALGEBRICO DEL CONTENUTO DEI DEPOSITI TRAN E MIN
	SAL	B	ST2	SE IL RISULTATO DEL CONFRONTO É ≥ SALTO ALL'ISTRUZ. ST2
	TRA		MIN,TRAN	TRASFERIMENTO DEL CONTENUTO DI TRAN IN MIN
ST2	ADE		I,UNO	I + 1 → I
	CAL		I,TNT	CONFRONTO DEL VALORE DI I CON '30'
	SAL	D	ST1	SE IL RISULTATO DEL CONFRONTO É ≤ SALTO ALL'ISTRUZ. ST1
	OAL		MIN	STAMPA DEL CONTENUTO DEL DEPOSITO MIN
	ALT	ØØ	ØØØ2	FINE DELLA ELABORAZIONE
	END			FINE DEL PROGRAMMA

Fig. 7.7 - Esempio 2 - Programma

APPENDICE

Semibyte di destra		Semibyte di sinistra															
	ES.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ES.	BIN.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000							0	ƀ	|	+						
1	0001							1	A	J	/						
2	0010							2	B	K	S						
3	0011							3	C	L	T						
4	0100							4	D	M	U						
5	0101							5	E	N	V						
6	0110							6	F	O	W						
7	0111							7	G	P	X						
8	1000							8	H	Q	Y						
9	1001							9	I	R	Z						
A	1010							[	&	-	_						
B	1011							#	.	$	'						
C	1100							@	]	*	%						
D	1101							:	(	)	=						
E	1110							>	<	;	"						
F	1111							?	\	,	!						

Tavola 1 - Alfabeto del Calcolatore C.N.D.

ISTRUZIONI	COD. SIMB.	FORMATO	LUNGHEZZA	FORMATO MACCHINA					
				1° BYTE	2° BYTE	3° BYTE	4° BYTE	5° BYTE	6° BYTE
ADDIZIONE DECIMALE	ADE	A	6	A 1	L1 L2	0	I1	0	I2
SOTTRAZIONE DECIMALE	SDE	A	6	A 2	L1 L2	0	I1	0	I2
ADDIZIONE ESADECIMALE	AES	A	6	A 3	L1 L2	0	I1	0	I2
SOTTRAZIONE ESADECIMALE	SES	A	6	A 4	L1 L2	0	I1	0	I2
TRASFERIMENTO	TRA	A	6	A 5	L	0	I1	0	I2
CONFRONTO LOGICO	CLO	A	6	A 6	L	0	I1	0	I2
CONFRONTO ALGEBRICO	CAL	A	6	A 7	L1 L2	0	I1	0	I2
INTRODUZIONE ESADECIMALE	IES	B	4	B 1	L	0	I1	—	—
INTRODUZIONE ALFANUMERICA	IAL	B	4	B 2	L	0	I1	—	—
ESTRAZIONE ESADECIMALE	OES	B	4	B 3	L	0	I1	—	—
ESTRAZIONE ALFANUMERICA	OAL	B	4	B 4	L	0	I1	—	—
SALTO	SAL	B	4	B 5	M 0	0	I1	—	—
SALTO CON MEMORIZZAZIONE	SME	B	4	B 6	0 0	0	I1	—	—
ARRESTO	ALT	B	4	B 7	messaggio			—	—

Tavola 2 – Repertorio base istruzioni macchina

ISTRUZIONI	COD. SIMB.	FORMATO	LUNGHEZZA	FORMATO MACCHINA			
				1°BYTE	2°BYTE	3°BYTE 4°BYTE	5°BYTE 6°BYTE
ADDIZIONE DECIMALE	ADE	A	6	A 1	L1 L2	I1	I2
SOTTRAZIONE DECIMALE	SDE	A	6	A 2	L1 L2	I1	I2
ADDIZIONE ESADECIMALE	AES	A	6	A 3	L1 L2	I1	I2
SOTTRAZIONE ESADECIMALE	SES	A	6	A 4	L1 L2	I1	I2
TRASFERIMENTO	TRA	A	6	A 5	L	I1	I2
CONFRONTO LOGICO	CLO	A	6	A 6	L	I1	I2
CONFRONTO ALGEBRICO	CAL	A	6	A 7	L1 L2	I1	I2
INTRODUZIONE ESADECIMALE	IES	B	4	B 1	L	I1	—
INTRODUZIONE ALFANUMERICA	IAL	B	4	B 2	L	I1	—
ESTRAZIONE ESADECIMALE	OES	B	4	B 3	L	I1	—
ESTRAZIONE ALFANUMERICA	OAL	B	4	B 4	L	I1	—
SALTO	SAL	B	4	B 5	M O	I1	—
SALTO CON MEMORIZZAZIONE	SME	B	4	B 6	O RI	I1	—
ARRESTO	ALT	B	4	B 7	messaggio		—
TRASFERIMENTO MEMORIA-REGISTRO	MRE	B	4	B 8	L RI	I	—
TRASFERIMENTO REGISTRO-MEMORIA	RME	B	4	B 9	L RI	I	—
ADDIZIONE REGISTRO-MEMORIA	ARM	B	4	B A	L RI	I	—
SOTTRAZIONE REGISTRO-MEMORIA	SRM	B	4	B B	L RI	I	—
CONFRONTO REGISTRO-MEMORIA	CRM	B	4	B C	L RI	I	—

Tavola 3 - Repertorio esteso istruzioni macchina

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

Tavola 4 - Potenze di 2

BYTE				BYTE				BYTE				BYTE			
BITS 0123		BITS 4567		BITS 0123		BITS 4567		BITS 0123		BITS 4567		BITS 0123		BITS 4567	
Hex	Decimale	Hex	Decimale	Hex	Decimale	Hex	Decimale	Hex	Decimale	Hex	Decimale	Hex	Decimale	Hex	Decimale
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	268,435,456	1	16,777,216	1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	536,870,912	2	33,554,432	2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	805,306,368	3	50,331,648	3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	1,073,741,824	4	67,108,864	4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	1,342,177,280	5	83,886,080	5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	1,610,612,736	6	100,663,296	6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	1,879,048,192	7	117,440,512	7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	2,147,483,648	8	134,217,728	8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	2,415,919,104	9	150,994,944	9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	2,684,354,560	A	167,772,160	A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	2,952,790,016	B	184,549,376	B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	3,221,225,472	C	201,326,592	C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	3,489,660,928	D	218,103,808	D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	3,758,096,384	E	234,881,024	E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	4,026,531,840	F	251,658,240	F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
8		7		6		5		4		3		2		1	

Tavola 5 - Tabella per la conversione esadecimale/decimale

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Tavola 6 - Tabella per le addizioni esadecimali

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Tavola 7 - Tabella per le moltiplicazioni esadecimali

M. ITALIANI - G. SERAZZI - LA PROGRAMMAZIONE DEL CALCOLATORE C. N. D.